

决策知识自动化

大数据时代的商业决策分析方法

Knowledge Automation

How to Implement Decision Management in Business Processes

【英】Alan N. Fish / 著　王飞跃 王晓 郑心湖 等 / 译

以具体的应用实践方式，展现知识自动化基础上的决策管理、业务规则及预测分析。

中国工信出版集团　人民邮电出版社
POSTS & TELECOM PRESS

数字版权声明

Alan N. Fish

FICO决策方案部首席顾问，使用业务规则和预测分析进行决策管理领域的权威专家，对象管理组织决策模型与符号制定委员会的联合主席。从事决策系统的构建工作已经有30余年，主持过许多当时的技术前沿项目。

王飞跃

中国科学院自动化研究所复杂系统管理与控制国家重点实验室主任，国防科技大学军事计算实验与平行系统技术研究中心主任，中国科学院大学中国经济与社会安全研究中心主任，青岛智能产业技术研究院院长。主要研究领域为智能控制、社会计算、平行系统、知识自动化等。

王晓

中国科学院大学研究生，研究方向主要集中于社会网络分析、动态网群组织分析与建模、社会交通、知识自动化。

郑心湖

美国明尼苏达大学（双城分校）研究生，研究方向主要集中于社会网络分析、智能交通、数据挖掘、机器学习、知识自动化。



决策知识自动化

大数据时代的商业决策分析方法

Knowledge Automation

How to Implement Decision Management in Business Processes

【英】Alan N. Fish / 著　王飞跃 王晓 郑心湖 等 / 译

人 民 邮 电 出 版 社
北 京

图书在版编目（ＣＩＰ）数据

决策知识自动化 ： 大数据时代的商业决策分析方法 /
(英) 菲什 (Fish,A.N.) 著 ; 王飞跃等译. -- 北京 :
人民邮电出版社, 2016.1
ISBN 978-7-115-40261-5

Ⅰ. ①决… Ⅱ. ①菲… ②王… Ⅲ. ①商业经营－经
营决策 Ⅳ. ①F715.1

中国版本图书馆CIP数据核字(2015)第204652号

内 容 提 要

本书综合大量来源于业务流程自动化的主流应用场景，聚焦于组织管理及运营中经营决策的知识自动化这一主题，向读者展示如何在实践中应用知识自动化技术实施决策管理，以提高运营效率和组织收益。

本书适合各企业 CEO、CIO、IT 架构师以及一切对知识自动化理论感兴趣的读者。

◆ 著　　　　[英] Alan N. Fish
译　　　　王飞跃　王　晓　郑心湖 等
责任编辑　朱　巍
责任印刷　杨林杰

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　　　　　　印刷

◆ 开本：700×1000　1/16
印张：11.25
字数：170千字　　　　2016年 1 月第 1 版
印数：1－4 000册　　　　2016年 1 月北京第 1 次印刷

著作权合同登记号　图字：01-2015-0739号

定价：49.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京崇工商广字第 0021 号

中文版序

当看到自己的著作被一种我并不会讲述的语言翻译而成的文字时，心中升起了一种意想不到的喜悦。我想这表明它已经满足了可被广泛交流的某些首要需求。这使我倍感荣幸。与此同时，想到自己的著作将被成千上万深受一种博大精深文化——正是这种文化塑造了令我肃然起敬的儿时东方神话和 21 世纪经济巨人——熏陶的智慧的人们阅读，内心也是极为忐忑。

中国科学院自动化研究所复杂系统管理与控制国家重点实验室的王飞跃研究员看到本书之后，决定将其翻译为中文。对此，我向他以及他的学生们——王晓、郑心湖、林懿伦、白天翔和王杰，表示由衷的感谢。他们出色地完成了对一份棘手的材料的翻译工作。我的一位中国同事 Jeremy Chen（FICO 的高级管理人员，技术产品经理），同时也是这一领域的一位专家，告诉我："译文的准确性以及简洁性令我印象十分深刻。"

最后，祝您在"知识自动化"中一切顺利。

There is an unexpected pleasure in looking at the translation of my words into a language I do not speak. I suppose it satisfies some primal need to communicate widely. Yet it humbles and terrifies me to think that my work will be read by thousands of intelligent people from an ancient and complex culture – the mythical orient of my childhood and the economic giant of the 21st century – which I hold in great respect. I thank you for reading this book, and I hope that the simple idea in it will be useful to you. As I say in Chapter One, useful ideas promote growth and prosperity, however prosaic they are, so long as they are widely shared.

It was Fei-Yue Wang, Professor and Director of The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing , who decided to translate this book into Chinese. My heartfelt gratitude to him and his students: Xiao Wang, Xinhu Zheng, Yilun Lin, Tianxiang Bai, and Jie Wang. They have done an excellent job with difficult material. My colleague Jeremy Chen (Senior Manager, Technical Product Management, FICO), who is a native Chinese speaker and an expert in the field, tells me that “the translation really impresses me with its fidelity and conciseness”.

Wishing you every success with your Knowledge Automation,

Alan N. Fish

译者序：迈向决策知识自动化

2012年底，我与几位同事赴美考察3D打印与社会制造产业，其中美国东部的纽约市是主要考察地点，因为新兴的几个小公司Makerbot、Shapeways、Quirky等都在那里。恰好，女儿也在那里工作，而她的朋友正是Shapeways的一位负责人，故请她帮忙联系部分访问事宜。当时正在纽约上州伦塞利尔工学院（RPI，也是我的母校）工作的张清鹏博士，也与夫人驱车数小时从Troy赶来相会，并请我和女儿到Hudson河畔晚餐。清鹏是我在美国毕业的最后一名博士，博士论文题目就是"人肉搜索"和众包机制，也是多年前我为实证社会计算研究而布置的起步工作。晚饭期间，我谈起自己刚刚完成的《从社会计算到社会制造：一场即将来临的产业革命》一文，讨论到关于社会信号的工作以及社会信号与知识自动化同物理信号与工业自动化的联系时，感到应当把知识自动化单列出来，不再局限于算法自动化，使其尽快成为一个独立的研究领域。女儿因在公司负责系统与运营，说看过一本新书，好像就叫"知识自动化"。

随后我立即在网上搜索，很快就找到了Alan N. Fish博士的*Knowledge Automation*，刚刚出版。浏览之后，觉得虽然内容与我们讨论的相差较远，但不失为决策知识自动化方向的一部优秀著作，特别是此书具有很强的实践背景。当时，我就动了将此书译成中文的念头，作为知识自动化的起步和普及材料。只是*Probabilistic Graphical Models: Principles and Techniques*的翻译工作还没有收尾，自己的学生和同事已因此耗费了太多的时间与精力，所以，尽管此书不厚，但也实在无精力和体力再开始新的翻译工作。我曾试着问女儿，是否有兴趣翻译此书。她想了几

天，表示不行，一是工作太忙，二是自幼长在美国，觉得中文基础难以胜任。

回国后，我得知侄女要去美国留学，就到自己过去工作的大学和系读研究生，而且准备选我曾建议并帮助创建的工程管理专业，与知识自动化的关系十分密切，因此又动员侄女来翻译 Alan 的书，并保证提供帮助。侄女答应了，还打算选过去的同事做导师，以交通知识自动化为题做论文，但因读书时间比较紧张，翻译工作一直没有开始。

没有想到，一份突如其来的海外趋势发展报告极大地推进了此书的翻译进程。2013 年 5 月，麦肯锡全球研究所发布题为《颠覆技术：即将变革生活、商业和全球经济的进展》的报告，预测了 12 项可能在 2025 年之前决定未来经济的颠覆性技术，其中代表“知识工作的自动化”之智能软件系统位居第二，列于“移动互联网”之后，“物联网和云计算”之前。这一分析预测使得“知识自动化”就是“知识工作的自动化”的认识一时风行于整个世界，引起业内外的一片热议。

在国内，麦肯锡报告引起了国家自然科学基金委员会和大学科研机构的许多专家学者的高度重视，并组织了相应研讨会。大家一致认为，知识工作的自动化是知识自动化的一种具体而重要的形式，是自动化学科发展的一个十分重要的新方向，是时代的要求，也是自动化走向智能化的必经之路。为此，我还将中国《自动化学报》创刊 50 周年专刊序文命名为《天命维新：迈向知识自动化》并在当年秋天的中国自动化大会做了题为“面向人机物一体化 CPSS 的控制发展：知识自动化的挑战与机遇”的大会报告，希望充分引起相关学者的重视。今天社会对智能技术的关注程度，也说明当时的认识是正确的。

2014 年春，新智元创始人杨静女士和图灵公司武卫东总编来我办公室，协商开办人工智能微信群事宜，我正负责智能自动化的科普传播工作，立即表示大力支持。同时询问武总是否有兴趣出版 *Knowledge Automation* 一书的中文版，因为感觉“图灵”的名字和带来的样书很适合本书的发行。武总十分热心，并很快办好了著作版权事宜。我的博士生王晓和其他几位刚刚加入团队的研究生和访问学生对翻译此书也表示了极大的热情和兴趣。在大家的共同努力下，本书的翻译，终于圆满完成。

回顾知识自动化的历史，起源可以追溯到古希腊亚里士多德的描述性知识和原始的形式逻辑以及工业革命之初英国布尔对其变革的拓展。半个世纪前人工智能的正式提出，以及随后而来的知识表示、专家系统、知识工程、智能系统等，构成了知识自动化完整的进度发展历史。知识自动化与人工智能中的知识表示和知识工程的关系是显而易见的，但其与专家系统的更加内在且深刻的关系，随着专家系统近乎彻底的消失而被一些现代学者所忽略。

本书以一种十分具体的应用实践方式，将专家系统的思想带回知识自动化。当然，我们应以一种新的、时代的、更高的眼光重视知识自动化的技术基础。一种可能的视角就是以智能算法和软件机器人为基础，针对 Cyberspace 中许多基于特定领域的“软件”任务，开发面向知识工作自动化的“知识机器人”。

本书作者 Alan N. Fish 博士是 FICO 公司决策方案领域的首席顾问，同时担任对象管理组织（Object Management Group，OMG）决策模型与符号（Decision Model & Notation，DMN）制定委员会的联合主席。自 2012 年 *Knowledge Automation* 一书出版以来，本书所提的理念与方法就得到了众多同行的认可，并在各个领域得到了广泛应用。特别是文中第 4 章所建议的 DRD（决策需求图表），现已被形式化为一种开放的 OMG 标准——DMN。DRD 的核心在于：通过结构化的描述决策、决策间的依赖以及决策所需的信息及知识，为参与知识自动化项目的所有人员提供了一个一致的、“带有预见性的自上而下”的框架。DMN 是作为业务流程建模符号（Business Process Modeling Notations，BPMN）的“姊妹”标准引入的：用户可以使用 BPMN 中的决策点来对业务流程建模，然后使用 DMN 来对这些决策点上执行的决策机制建模。对于 DMN 标准及其与 DRD 的区别，有兴趣的读者可以参考作者与人合著的 *Emerging Standards in Decision Modeling：An Introduction to Decision Model and Notation* 一书，或登录网址 http://www.omg.org/spec/DMN/ 查看。截至本书出版这一刻，市场上已经存在许多 DMN 建模工具了，这些工具可直接绘制 DRD，定义或引入潜在的决策逻辑，并将生成的决策模型部署为可执行的决策服务。作为 DMN 标准的主要制定者之一，Alan 一直活跃在决策管理与决策自动化领域，并引领着这一领域的发展。

为什么我们现在需要知识自动化？一言以概之：工业时代需要工业自动化，知识时代必须知识自动化。工业时代的发展在许多方面对人类的体力提出了“非分”的要求，迫使人们必须依靠工业自动化的手段来“补偿”其体能上的不足，才能实施、运营、维护各类大型或精密的系统和过程；同理，面临物联网、大数据、云计算、智能技术等，正在迅速兴起的知识时代也对人类的智力提出了更高、更加“非分”的要求，人们更需要借助知识自动化的方法来“弥补”其智能上的不足，进而才能去完成各种层出不穷的不定、多样、复杂任务。

大数据特别是社会信号的出现，为知识时代复杂任务的描述、建模和解析提供了新的思路。以网络信息为主要表现形式的开源数据，虽然被认为是印刷出版发明后知识之产生与传播的又一场革命，但其价值仍未被充分地认识，在过去相当长的时间里，对于正式渠道的态势评估和决策支持而言，基本还处于“无用”状态。无用之用，众用之基：大数据时代，我们应该以开源信息为主，汇集海量数据，然后通过定量的方式来描述、分析、评判知识的价值，使之服务于企业、科技甚至社会决策。

从技术发展的角度而言，在机械化、 电气化、 信息化、 网络化之后，我们正进入第五个发展阶段，即虚实互动的平行化时代。我们认为，软件定义的系统、“激活”的情报以及灵活的动态网群组织（CMO）将成为平行时代成功企业管理和运营不可或缺的关键因素。软件定义系统的构建和使用，为企业管理和决策提供了真实可靠的科学依据。在企业内部，构建与之互动演化、反馈执行、协同发展的人工流程、人工工厂及人工企业，可针对流程执行管理、资源分配调度及企业生产制造进行建模计算、生产优化以及监控预警。在企业外部，利用知识自动化技术，可针对特定问题，将各种各样的相关网站、博客、论坛、微博、微信及其组合视为观察和获取企业决策所需知识的“信源频道”和“传感网络”，从而高效并准确捕获目标定位的企业竞争情报和决策需求信号。随后基于软件定义的企业对需求信号进行数据计算和知识解析，构造“激活”的情报，为企业流程管理、生产制造、决策制定提供有效、准确和灵活的输入。其中，CMO是社会信号获取、社会需求定位的关键。这要求我们必须充分认识到世界正从牛顿系统为主导向以

默顿系统为主导的演化，并迫切需要考虑智能的“虚实二象性”。在牛顿系统中，多数定律是发现的；在默顿系统中，多数定律是发明的。利用泛在的便携式移动传感设备、蓬勃发展的社会媒体以及网络世界无限的信息和智力资源，知识自动化将把人纳入数据生成、信息感知、知识推理、智慧解析、决策执行，然后生成新的数据的整个闭环过程中。因此，平行时代的知识自动化必然是信息化、自动化、智能化、人机化等的有机融合，必须从物理信息系统（CPS）走向物理社会信息或社会物理信息系统（CPSS）。

十分感谢为本书翻译做出贡献的各位学生和同事，他们是王晓、郑心湖、林懿伦、白天翔、王杰、袁勇等，同时，王成红教授、桂卫华院士、柴天佑院士、孙优贤院士、郑南宁院士、张钹院士、黄琳院士、吴宏鑫院士以及多位参与基金委“知识自动化”研讨会的专家和学者也在本书的翻译过程中给予了大力的支持，并对译稿提出了宝贵的修改意见。Alan 本人也提供了很多帮助并充分肯定了书稿的翻译质量，在此表示诚挚的感谢。最后，向新智元创始人杨静、图灵公司武卫东总编和朱巍编辑及其同事在本书翻译过程中所提供的帮助深表谢意。

希望本书的出版，能够有效地促进知识自动化和智能产业的发展和深入。

王飞跃

2015 年夏秋记于青岛、长沙、北京

序

初次接触 Alan 的工作是在 2009 年，当时他向我介绍了决策需求分析的核心理念。他所描述的方法以及该方法在改进决策管理系统的分析和设计方面所蕴含的巨大潜力，给我留下了极为深刻的印象。过去十年里，大多数时间我都在从事决策管理方面的工作，帮助企业使用业务规则和预测分析技术来自动化并改善其业务决策。Alan 提出的收集、建模和管理决策需求的方法，立刻使我意识到，这恰恰是改善业务决策的正确方法。从那以后，我一直使用该方法来帮助我的客户。

决策管理是一种对经营业务决策的自动化和改进都行之有效的方法，这里的决策包括那些影响单个客户或案例的微观决策。在我从事决策系统管理工作的这些年里，这一方法逐渐为大众所知并接受。要构建一个决策管理系统，首先需为决策流程逻辑的管理搭建一个坚实的平台——业务规则管理平台，而且还要具备有效整合该逻辑与预测分析模型的能力。同时，该平台还要求使用以决策为中心的方法，能有效识别、建模、提取和管理需求。

以我最近合作的一家企业为例。当时他们正尝试将一套业务规则管理系统应用到企业以过程为中心的文化中去。运用 Alan 的方法，我们将精力集中在决策、决策间的依赖以及决策所需的信息与知识方面。最终，业务规则管理系统成功上线，该企业对决策在业务过程中的作用也有了新的认识。

许多企业错误地认为，他们现有的需求收集与流程管理同样适用于新的业务规则及其管理系统。但实际上，由于旧的方法缺少对决策的关注，因此对分析决

策几乎没有帮助，而这恰恰又是运行成功的必要保障。类似地，正在使用的这些预测分析模型往往也缺乏一个正式的或可重用的方法来定义它所要辅助的决策，而这正是决策需求分析的价值所在。

本书的出版正当其时。目前，决策管理系统越来越受人瞩目；同时，业务规则和预测分析产业正将其关注点转向对于决策和决策自动化的讨论。过去九年里，我帮助过几十家实施业务规则和预测分析技术进行业务管理的企业，并与超过上百家企业谈论过这一话题。对企业和组织而言，决策管理系统能够有效地改进其业务，并使业务更为敏捷、更易于分析且更具适应性。若想获得成功，企业不仅需要强大的技术，更需要有能力分析和设计自身使用的系统。Alan 在决策需求分析方面的工作正是使企业具备这一能力的关键。

在本书中，针对决策需求分析方法的使用，Alan 向读者提供了一个简单且实用的指南。同时，他对知识、决策机制及不同场景下业务流程中决策服务的关键角色进行了简短而全面的介绍。本书核心章节主要介绍了决策需求的建模方法以及相关研讨会。从发现问题到实施决策，Alan 展示了该决策方法是如何对项目每一阶段的实施提供帮助的，同时，本书还提供了大量常见决策模式及其案例分析，以便帮助读者快速了解掌握这种方法。如果你计划对决策进行分析和建模，这本书就不容错过，它将引领你顺利完成工作。

我强烈地认为，本书所描述的方法是目前应对决策建模问题的最佳方案，这也是我使用它来服务我的顾客并在我的新书里极力推荐的原因。

——James Taylor

James 为决策管理解决方案公司（Decision Management Solutions，位于加州 Palo Alto）的 CEO 和首席顾问。著有《决策管理：商业规则与预测分析指导手册》一书（*Decision Management Systems: A Practical Guide to Using Business Rules and Predictive Analytics*，IBM Press，2011）。作为一名活跃的咨询师，他与全球许多企业保持合作关系。

James 的联系方式为：james@decisionmanagementsolutions.com。

前　　言

许多组织一直在寻找能够使他们的业务流程更为高效、低耗并便捷的方法。近年来，常用的策略多包含以下 3 种技术。

(1) 决策管理：运用预测分析方法与业务规则对决策制定流程进行优化与自动化，使得日常经营策略可为流程管理人员而非 IT 人员直接掌控。

(2) 业务流程管理系统（Business Process Management Systems，BPMS）：此类系统使用序列化的活动对业务流程进行建模，并通过执行模型来保证流程合乎规定。

(3) 面向服务的架构（Service-oriented Architecture，SOA）：一种软件组件模型，提供松耦合的服务以及可重用的功能组件，对内部逻辑进行封装，从而实现对用户透明。

本书着重强调这些技术领域的交叉融合。综合运用上述三种技术，就可以在服务中封装业务知识。BPMS 通过调用这些技术封装了业务知识的服务，自动为组织做出最优经营决策。这一过程即为知识自动化。（见图 0-1。）

流程自动化项目通常是庞大且复杂的，它往往被组织视为一种重要的战略投资。当部署完所需的 IT 设施之后，组织还将面临流程重设计和组织结构变动等挑战。若自动化的范围包括经营决策的制定流程，我们还将面临新的任务：创建执行决策的服务。这些“决策服务”可能非常复杂，包括各种高级的分析模型、成百上千的业务规则以及大量的计算工作等。

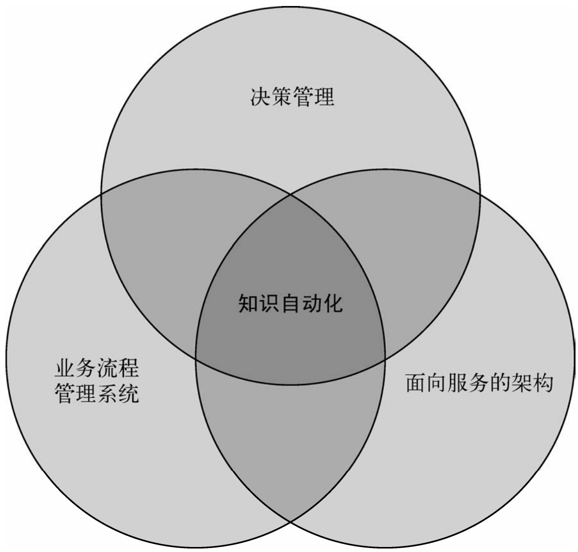


图 0-1　知识自动化

因此，若想要获得成功，需对知识自动化项目进行精心管理。项目可能出现的问题包括：

- 开发项目出现延迟交付或是严重超支的情况；
- 设备因其性能不能充分满足特定需求而被用户拒绝；
- 系统难以维护，且混乱程度逐年增加。

之所以出现这些问题，往往只是由于在项目前期，没有对决策的需求做出清晰定义所致。在我看来，这些问题的存在并非由于不够努力，而是由于采用了错误的方法。那些失败的项目，往往在一开始就已经出错了。

许多书讨论了决策管理、业务规则以及预测分析的理论和技术背景，并对这些技术的用途做出了清晰的解释（本书最后附有参考书目）。我也将简单介绍这些技术及其益处的一些基本知识，但这并非本书的核心内容。我假设读者已经参与或使用了一个知识自动化项目，或至少对于决策管理的优点有所了解，并在考虑这样一个项目的可行性及其规划。本书的主要目的是为知识自动化项目的逐步实施提供一个实用的指导手册。

这些年来，我参与了众多业务领域决策服务构建相关的大量项目。或许是由于本性比较懒惰，我开发了一套可以简化项目并降低项目失败和超支风险的工作方法。人们往往会低估简单这一美德的价值：我们选用的方法，应使人们可以更加容易地完成任务。当一个任务的范围可被清晰界定时，人们往往会认为该任务易于完成，因为他们清楚地知道自己该干些什么；同时，一个清晰定义的任务结构也更易于被分解为更小的子任务。

书中描述的方法为决策需求分析（DRA）法，该方法用于定义将要自动化的决策流程的范围和结构。DRA 快捷易用，且能清晰地定义业务决策制定流程的覆盖领域，因为：

- 它将决策制定与业务流程中的特定点——决策点——绑定在一起；
- 它基于一种简单的描述决策制定流程结构的图表——决策需求图表（DRD）；
- 它允许以罗列决策点、在 DRD 中绘制边界的方式来定义决策制定流程的范围。

因此，DRA 表述的需求首先是一张图表，这是一种可被所有项目参与人员，包括项目经理、流程专家、业务分析师、技术架构师、开发小组等都轻易理解的通用语言。DRD 使得决策流程自动化的需求在项目伊始就可被所有人讨论并得以确定。同时，DRD 揭示的决策制定流程结构可直接用于项目本身的设计、实现等。希望读者能发现它的有用之处。

如何阅读本书

首先完整地通读一遍。不用担心，本书的篇幅并不长，也未纠结太多技术细节。读者通常希望作者会针对不同类型读者提供多种可选阅读途径，但本书建议使用的方法基于这样一种思想：DRD 展现的决策制定流程结构为参与项目的各类人员都提供了一个自上向下的框架。因此，想要完全理解这一框架的价值，需要对该方法有一个全局的了解。本书的组织结构如下。

第 1 章介绍了“知识的价值”这一概念，首先在宏观经济层面阐明了知识对于经济增长的贡献，然后介绍了它对个体组织的独特价值。这一章对决策管理的基本原则进行了简要描述，并介绍了一种衡量决策自动化的效益的方法——决策收益定价法。

第 2 章讲述了如何重新设计业务流程以使 BPMS 可在关键点上调用决策服务来实现经营决策自动化。本章给出了一种业务流程合理化的方法，可使决策管理中的收益最大化，同时提供了一个简单的贷款发放流程模板作为示例。

第 3 章描述了在决策服务中封装业务知识所需要的最为重要的几项技术（如业务规则、算法以及预测分析等）及其主要内容，并解释了这些技术的适用场景。然后阐明了如何使用由 BPMS、预测分析模型和决策服务等构成的周期循环架构进行决策管理。

第 4 章阐明了决策需求分析的基本原则，并介绍了决策需求图表（DRD）。DRD 可将一个决策域描述为由决策、支撑决策的知识和数据构成的网络。本章详细描述了如何通过组织一个多方参与的研讨会，生成 DRD 并为决策自动化的项目范围文档提供所需的支持信息。

第 5 章演示了如何利用 DRD 的结构对项目进行范围界定、评估、规划及管理，以此实施一组决策服务。本章内容还展示了 DRD 的结构在决策服务设计、开发、配置和测试中的基础性作用，它如何使所有活动有效组成一条“知识生产线”，并允许所有项目交付组件都可以全程追溯到其最初的原始需求。

第 6 章提供了一组常见的决策模式及其简化版局部 DRD 和对象模型，此类模式通常被作为模板使用。最后，本书还考虑了这样一种情况：决策服务与业务用户需要协同合作才能达成一个决策，并提供了一些备选处理方案及其适用场景。

目标读者

一个知识自动化项目的运作需要多方工作人员在一系列级联活动中的积极参与，希望本书能够对他们有所助益。

- 组织中的高层管理者，尤其是 CEO、CIO 和 CFO。在考虑或实施决策机制自动化时，他们需要了解实施项目所需的技术及其基本原理，以及该过程中可能出现的问题。
- IT 架构师，负责采购 BPMS、BRMS 及其他分析设备，他们需要了解设备可为知识自动化提供的功能以及如何实现设备之间的交互操作。
- 分析师，负责设计或重设计业务流程以使其实现决策自动化，他们需使用有力的手段来保障清晰而有效的流程设计。
- 人事管理部门，需要了解引入知识自动化系统后人员职能的变更，并需根据这一变化做出人事部署上的调整。
- 组织管理人员，特别是自动化项目重点关注的那些组织性能方面的负责人员（如产品经理，风险管理人员，流程管理者等），需确保新系统能够更为高效敏捷地支持其部门工作。
- 产品经理，需要了解如何将一个知识自动化项目结构化并加以管理，以使其费用开销、时间开销和风险最小化。
- IT 架构师，参与自动化决策系统的设计，需要一套基本原则来明确一系列决策服务，并以此设计内部结构和接口签名。
- 分析师，负责定义自动决策系统的业务需求，使用一种自上向下的方法来发现并编写业务知识。

同时，我也希望本书能够引起那些更关心知识自动化理论的读者的兴趣。包括：

- 各行业的标准机构及组织，通常他们想要定义可执行的决策模式，其中涉及多种形式的业务知识；
- 某些科研机构，他们通常会开设 IT 和 AI 课程，并希望向学生讲授一种强有力的知识自动化方法；
- 一般读者，例如有些人希望了解金融产品申请背后的流程，想了解金融机构是如何做出自动化决策，从而直接影响到他们的生活。

本书内容

本书是一本有关“如何去做”的实用指南而非理论性著作，主要面向知识自动化项目工作的参与人员。所以，本书严格遵循流程自动化的普遍范式：BPMS 和 SOA。特别地（第 2 章将会详细讲述这一观点），书中假设无状态的决策服务将在 BPMS 管理的业务流程中被调用。因此，本书有意忽略了一些新兴领域，如复杂事件处理（Complex Event Processing，CEP）——在 CEP 中，决策服务不是由业务流程中的一个任务触发，而是检测到一个事件就触发的。不过，本书的核心观点——聚焦如何表示和执行自动化决策制定流程结构——则适用于任何一种决策模式。

本书试图保证讲述的内容是与产品无关的，因此着重描述系统功能而不过多涉及技术细节。书中不会对比不同 BRMS 产品的功能优势，亦不会对 BRMS 中的推理过程或对支持 BPMS、BRMS 的硬件系统架构进行讨论。因而，我希望本书描述的方法和技术能在任意产品和架构上都得到应用。

顾名思义，本书将聚焦企业业务流程中经营决策的知识自动化这一主题，而不会过多着墨于其他决策管理的应用（如产品设计与营销、商务智能、数据挖掘等）。本书的示例来源于流程自动化的主流应用场景（坦白地讲，这正是我最熟悉的领域），包括银行、小额信贷、普通保险、人寿保险和健康保险等领域的发放处理、账户管理、索赔管理等内容。当然，知识自动化也可以被用于许多其他的场景。

致　　谢

我要向许多人致以我最诚挚的感谢。

- 我的妻子 Cath，感谢她在我写书期间给予我坚定不移的支持，感谢我年幼的孩子 Oscar 和 Penny 对于我不能常伴他们左右的宽容和理解。
- 我在 FICO 的同事，感谢他们允许我醉心于实施书中涵盖的想法，并在业内进行推广，也感谢他们在工作中的积极配合。
- 我所有项目的客户，感谢他们对我的信任和友好，这使我的工作成为了一种享受，我从他们那里受益良多。
- 我的挚友 Roger Farmer，感谢他给我的建议、信心以及龙舌兰酒。
- James Taylor，感谢他对我的支持，他知识渊博，并欣然同意为本书作序。
- Paul Konnersman，感谢他参与这个漫长的通信过程，与我探讨决策自动化的方法，并允许我引用他的作品，即便他并不完全同意我的看法。
- 我已故的父亲，他教导我如何从系统的角度出发思考问题。我怀念与他争辩的时光。

我在维也纳一家最老的爵士酒吧——Jazzland 的砖拱地窖里草拟了这篇简介。同时，一边还吃着克罗地亚烤肠，喝着奥地利啤酒，观看着萨克斯手 Vincent Herring、键盘手 Danny Grisset、贝斯手 Markus Schieferdecker 和鼓手 Joris Dudli 的演奏，他们真是一支令人称奇的乐队。我必须承认，我极其荣幸能够在如此有趣的领域，从事一份我热爱的工作，并与睿智的客户和同事共事。

愿能永远如此。

目　　录

第 1 章
知识的价值

当听到“知识”一词时，你首先会想到什么？是古老大学图书馆里落满灰尘的旧书，还是身着白袍、额头宽大的男人？我希望阅读第 1 章内容能够改变你对知识的看法。知识并不是与商业世界毫无瓜葛的抽象概念，而是实实在在的公司财富。你可以生产它，拥有它，购买它或者销售它，甚至将它植入机器中为你创造财富：知识是具有价值的实体。这一概念是理解决策管理如何带来收益的关键所在。

在本章开篇，我将对现代宏观经济中有关知识的理论做一简短探讨，随后再从实践的角度讨论组织该如何衡量并发掘自身业务知识的价值。

1.1　经济学中的知识

在过去的半个世纪里，我们对经济发展动因的理解发生了革命性的变化。古典经济理论强调劳动力、土地和资本的重要性，而现代经济理论则更加重视人类知识及其对经济增长的重要贡献。

1.1.1　新古典增长理论

20 世纪 50 年代提出的增长理论，如 Robert Solow[1] 和 Trevor Swan[2] 的理论，几乎完全是基于对物质资本（如机器和股票等）的假设。此类理论被称为“新古典增长模型”。它假设物质资本服从边际递减效应：若持续增加某单项生产要素的

投入，达到一定水平之后，对应的单位产出会逐渐减少。这一假设用数学表示显得非常简洁，它使得经济模型将最终聚合为唯一的均衡状态。换言之，在其他条件不变的情况下，增长将逐渐减缓并最终停滞。

图 1-1 反映了这一观点在当时的主流地位。Bill Phillips 在 1949 年建造了一台液压计算装置 MONIAC，通过调节管道和水箱之间的水流来模拟英国经济。该模型复杂且精确，其传达的信息十分明确：经济可以自我调节。

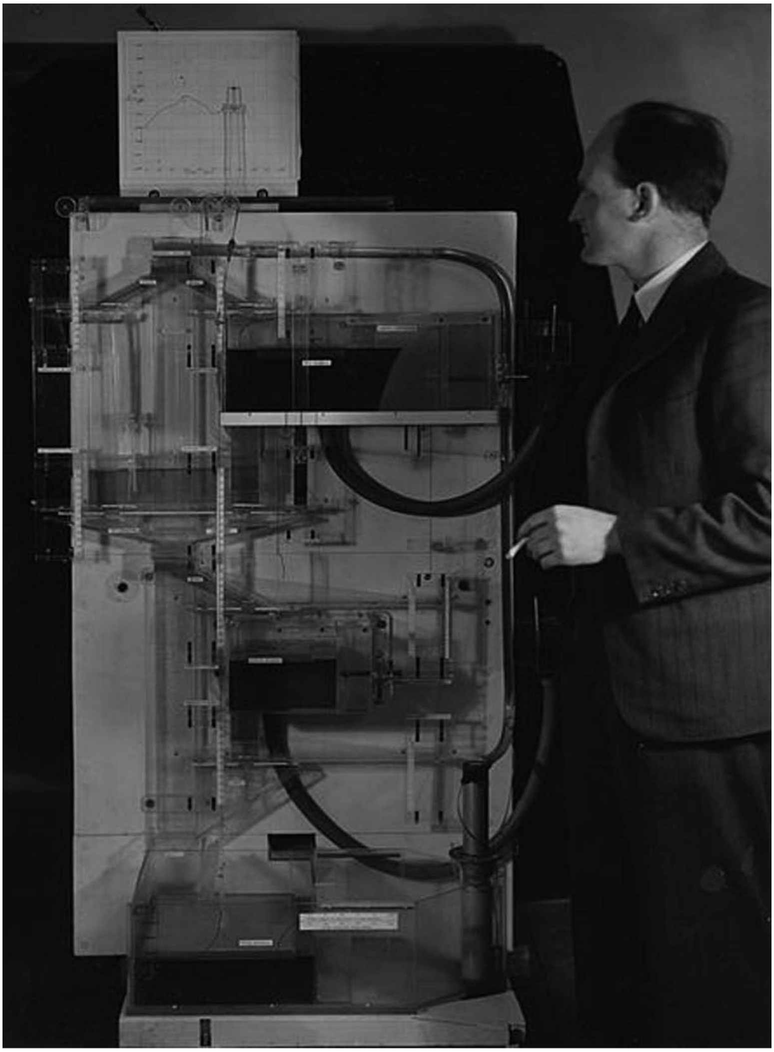


图 1-1　Bill Phillips 和 MONIAC

新古典模型无法也无意解释长期增长。在该理论框架下，长期增长是由外部影响所引发的。Robert Solow 将技术性知识作为一个“外生变量”[3]引入了新古典模型。他假设，在不受经济模型的影响下，技术会得到稳定进步，即技术为经济的持续增长提供外部激励。这一模型可用于衡量技术进步对于经济的影响，或用于解决如给定经济增长率条件下最优储蓄率之类的问题。因此，Solow 模型在经济分析领域得到了广泛的应用，他也因此获得了 1987 年的诺贝尔经济学奖。但是，这一模型并不能解决如何制定经济政策才能实现技术进步的问题。

1.1.2　新增长理论

因而，近代增长理论更倾向于将知识作为内生因素来处理：知识作为经济模型的内部变量并用来解释所观察到的经济增长现象。Robert Lucas 如此解释这类模型的重要性：

> 印度政府可否采取某些措施使得印度经济增长达到类似印度尼西亚或埃及的速度？如果可以，那么应该采用何种措施？如果不可以，那么究竟是哪些“印度特色”导致了这一结果？这类涉及人类整体利益的问题十分复杂：一旦开始思考这类问题，就很难再有精力去考虑其他事情了。
>
> 这正是我们需要一个新的经济发展理论的原因：为我们提供一个组织此类事实的框架，并判断哪些是偶然，哪些是必然[4]。

根据 Charles Jones 和 Paul Romer 的理论，为解释那些最重要的现象，“一个增长模型必须考虑到理念、体系、人口以及人力资本等因素之间的相互作用。当前经济增长的两个主要现象——伴随全球化发生的市场极度膨胀和长期以来经济加速增长——很容易用理念的典型特征来解释，这就是非竞争性[5]。”

竞争性的概念对于理解人类知识在经济增长中的作用至关重要。绝大部分的物质资本都是竞争性的：在一段给定时间里，只有一个人可以使用它们。因此，人们会为了土地、金钱或其他物质资本的产权展开竞争。然而，理念是非竞争性的，人们无需为之展开争斗。某一理念可被许多人同时使用而不会对其有所损耗；

而且，与物质产品恰恰相反，理念的价值与使用它的人数成正比。

因而，非竞争性物品并不会像物质资本那样遵循边际报酬递减规律，相反它们遵循的是边际报酬递增规律。例如，一种编程语言就是一类非竞争性物品。一个程序员使用一种编程语言并不妨碍其他人也使用它。事实上，如果有许多人都在使用同样的语言，那么这种语言的价值将会上升，而使用它的成本将会下降。正是非竞争性物品的这种规模效应导致了经济增长会不断加速，而不是最终趋向均衡。正如 Romer 在其斯坦福大学传记中所解释的那样：

> 我一直对增长率为何会不断上升感到好奇……过去的理论认为，随着人口的增加，物品的稀缺性将会越来越严重，但事实却并非如此。以新技术的形式呈现出的新理念是解释这一现象的关键。人人都“知道”这个道理，但新技术为何会出现得越来越快？一个十分重要的原因在于，理念是可共享、非竞争的，越多人参与就会使更多人受益。从这点看来，理念的特性与稀缺物品恰好相反。[6]

经济理论中，物品的另一重要特征为是否具有排他性。这一特性是指物品具有可以阻止人们使用该物品的特性，除非该物品已被他们购买。如表 1-1 所示，竞争性和排他性这两个要素可产生四种可能的物品类型。

表1-1　商品类型

	排他的	非排他的
竞争的	私人物品（食物、衣物、住宅）	公有物品（鱼类资源、木材）
非竞争的	俱乐部物品（社交俱乐部、卫星电视）	公共资源（空气、灯塔）

大部分人类知识都处于公共领域并被视做公共资源。但在自由市场经济的环境下，会产生这样的问题：若某物品可以被免费获得（非排他性）且其共享不会导致竞争（非竞争性），那么组织就难以通过提供该物品来获得收益。这也是为何基础科学的研究往往是由政府出资支持的原因。不过，知识也并非总是公共资源。两种机制的出现使得知识生产者可从中获利：知识产权（专利和版权）和保密制度（商业机密和保密协议）。在后续章节中我们将会看到一个信用风险评估的例子，

在这个例子中，与客户相关的知识可像俱乐部产品那样（如卫星电视）进行销售。

通过将理念和知识定义为非竞争性、部分排他性物品，Paul Romer[7] 和 Robert Lucas[8] 将知识和理念作为内生变量纳入了他们的经济模型，并提出了一种基于人力资本而非物质资本的新增长理论。该理论中，经济增长主要依赖以下两个因素：(1) 理念数量的一般性增长；(2) 理念的交互与扩散，包括购买知识并将其运用到企业运营之中。换言之，推动经济增长和整体利益增加的不再是劳动力或资本，而是源源不断的知识。

1.1.3　知识经济

经济学家常把知识比作一类制造工艺：他们将知识的增长看作是新物品生产能力的提升。这固然是相当重要的一方面；但另一方面，越来越多的企业，如整个金融产业，已经完全不生产任何物质产品了，他们只处理信息。这些产业的出现建立在信息技术的基础之上，信息技术使得人们可以直接利用知识并结合现有信息做出更好的决策。商业部门还未很好地参与全球经济增长这一过程，企业仍有潜力更好地利用基于知识的决策技术，挖掘新的商业机遇。正如 Tom Debevoise 所观察到的那样：

> 基于旧的经济模型建立的业务流程将无法为未来经济带来增长和利润……更微妙的是，没有基于知识的决策作为支持，业务流程甚至无法保证持续盈利。此外，大部分流程都基于一些过时的假设而构建，其表现并不稳定。“婴儿潮”一代留下来的知识对许多企业已不再适用。在各个领域以各种形式出现的“知识”以及“创造新知识”，成为所有现代企业赖以成功的关键。知识需要被识别、提取并被包含在决策当中，以便生成并维持敏捷的企业结构。这使得由知识驱动的业务流程可以生产出符合市场潮流的产品、服务，并因此盈利。[9]

知识经济的发展使得知识可被交易，而信息技术的发展使得基于知识的决策可被嵌入知识驱动的业务流程中，亦使得全球经济最终受知识的累积和扩散推动

而增长。但从另一方面看来，作为一种销售商品，大部分知识依然与最原始的燧石箭头一样，需要通过劳动密集型的流程来生产。大学与研究中心并非知识的工厂；它们只是小作坊而已，作坊中知识的生产率受限于单个知识工匠的生产力。

然而，随着机器学习（涵盖文本挖掘、规则演绎，以及第 3 章会详细介绍的预测性分析等内容）的发展，这一情况即将发生改变。从本质上来说，机器学习是一个从数据中自动创造知识的过程。在不久之前，这还是一门高度专业化、仅仅为一些统计专家所用的技术，但现在它已发展到能被一般商业用户所使用的程度。企业已可自动生成他们自己的知识，并运用这些知识来实现决策自动化。例如，一家信用卡公司通过分析他们现有客户的行为数据来生成预测模型，并使用模型自动决策新申请用户的贷款限额及利息率。此类模型是“非普适类”知识的一种形式，它们往往只针对特定市场、产品或者工作方式。公司采用这一方法所获得的竞争优势将通过知识保密的方式得以维持。

关于经济学的最后一点讨论，让我们回顾一下 Romer 的观点：

> 在所有理念中，最重要当属元理念。这是指那些能够支持其他理念产生和传播的理念。英国人在 17 世纪发明了专利和版权这两大制度。美国人在 19 世纪创立了现代研究型大学和农业推广服务，在 20 世纪发明了基于同行审议的基础科学竞争性拨款制度……我们还不知道下一个重要的元理念会是什么，或是它将在哪里出现。但有两点可以很明确地预知：首先，一个国家要在 21 世纪取得领导地位，必须对其私营企业的理念创新及其实现施以更加高效的支持；其次，关于第一点的元理念一定会产生。[10]

机器学习是 21 世纪最令人兴奋的元理念之一：从数据中自动生成知识。这一理念将对科学、政务以及商业等学科产生重要的影响，也将在自动业务决策制定中被广泛应用。这一技术将对提高增长速率产生什么样的影响还很难预知，但可以肯定，随着可用数据爆炸式的增长，其影响必将十分深远。

1.2　知识型业务

现在，我将专注于本书的核心主题——决策管理，这是当前蓬勃发展的知识经济中十分重要的一个方面。决策管理运用机器学习和自动决策技术来提高业务决策的盈利能力。知识不仅可以在宏观经济层面创造总体价值，同时可为个体企业创造特定收益，且这些收益是可被度量的。知识被用于业务决策之中，而这些决策将会影响业务收益。

1.2.1　战略决策与经营决策

James Taylor 指出，并非所有决策都具有相同的权重或价值，因此，有必要对它们进行分类。[11]

- 高价值决策可被称为*战略决策*。战略决策只能由董事会或高管成员制定。战略决策一般包括合并与收购、投资与撤资、裁员、市场定位等。这类决策对公司的影响往往以百万甚至千万美元计。
- 低价值决策可被称为*经营决策*。经营决策往往由与顾客接洽的一线员工制定。经营决策一般包括产品选择、交叉销售、向上和向下销售、业务申请的接受与拒绝等。一个经营决策的影响可能只有几美元，甚或几美分。

当然，在这两种极端之间还有一些中等价值的决策，涉及分析、研究、开发等由公司中层管理人员及专家团队制定的决策。这些中等价值决策包括产品研发、定价、市场营销等内容。

观察这些决策的价值、制定频率、所需的数据及知识等因素之间的关系，我们可以得到一些有趣的结论。

战略决策的价值很高，但这类决策的数量往往少之又少。董事会可能一个月才召开一次，一年可能只会做少数几次有重大价值的决策。此外，战略决策需要庞大的信息支撑。一次收购决策往往需要大量收集公司的报表及账务信息、产品描述、市场评估以及重要人员的详细资料等。要做出决策，执行团队还需使用复杂的财务和商业模型，同时结合个人多年的教育和亲身经历，以及公司法和会计法相关的知识。

与之相反，一线人员每天都需做出成千上万的低价值经营决策。每个经营策略一般只涉及单个案例或事件，通常直接使用从客户处收集到的信息（如申请表、索赔表等），因而此类决策的可用信息往往十分有限。同时，用于此类决策的知识也较少；经营策略通常严格遵从公司的规章或制度，对一线工作人员的训练也往往仅需几个月的时间。

表 1-2 列出了它们的关系。

表1-2　战略决策与经营决策

决策级别	单个决策价值	频率	所需数据量	所需知识量
战略决策	高	低	多	多
经营决策	低	高	少	少

从表 1-2 中我们可以得到两个十分重要的结论。首先，单个决策对于公司的价值反映了决策所需的知识量。其次，正如 Frank Rhode 首先提出的那样[12]，虽然单个经营决策的价值较低，但其整体价值却有可能与战略性决策相当。这意味着公司应给予经营决策和战略决策同等的重视。

后面的章节将会指出，任意一项决策实现自动化所需的投资几乎完全由该决策所需的知识量和数据量决定。其暗含的不同层次上决策的投资回报率也十分清晰：战略决策的投资成本会更高，而经营决策则较低。但经营决策自动化带来的回报可能会与战略决策自动化带来的回报一样多。回报与决策的频率相关，但投资成本却与频率无关，因为相同的业务知识可以被应用到所有情况中去。

因此，在决策管理中，经营决策的收益是唾手可得的；而战略决策却需要商务智能的支持：使用一系列技术对数据进行聚合、分析、过滤以及可视化以辅助管理层进行决策。

战略决策与经营决策

战略决策由高级管理层制定，其发生频率较低，单个决策的价值较高。这类决策往往涉及大量的数据和业务知识，因而其自动化难度很大且代价很高。

经营决策由一线员工制定，其发生频率高，单个决策的价值较低。这类决策所需的数据量和业务知识相对较少，因而其自动化难度较小且代价较低。

1.2.2　决策收益定价法

到目前为止，我一直在泛泛而谈价值这一术语。那么，具体而言，究竟如何利用自动化经营决策中封装的业务知识为企业创造价值呢？Fair Isaac 公司（FICO）研发了一套称为决策收益定价法的方法，可以预测和衡量决策管理项目的投资回报率（ROI），该方法从五个维度评价了决策机制的表现：精度、成本、速度、敏捷性和一致性。

> **精度：**市场营销及客户管理系统、客户服务中心、分支机构等部门执行决策的有效性。该维度可由一系列财务指标，如利润、客户终生价值、收益、损失以及成本与预期产出比等来衡量。
>
> **成本：**决策制定的效率，与执行决策的开销有关。通过业务相关开销、决策执行所需数据（如评分、信用咨询报告、机动车报告、人口统计资料）开销、系统资源开销以及其他开销（如邮件开销）等来衡量。
>
> **速度：**决策执行的速度。该维度可通过计算请求与应答时间间隔、周转期间客户损失数、批处理运行时间等指标来衡量。
>
> **敏捷性：**取决于在系统内部和组织架构内部更改决策策略的快速及方便程度。可通过以下三方面衡量：业务经理更改涵盖从项目设计、测试到实施的规则或策略所要花费的时间；改变决策策略需要新申请的资源数量；新信息多久可被运用到决策策略之中。
>
> **一致性：**决策在整个企业中被整合与协调的程度。可通过决策在不同时间、不同渠道的一致性以及在同一产品线内和不同产品线间的一致性来衡量。
>
> ——改编自“如何评估你的决策收益定价法”
>
> （How to Assess Your Decision Yield）
>
> FICO 高层简报 1961 EX（2009）

在执行决策管理之前和之后，一个决策流程在上述五个维度的表现被可视化为一张雷达图（如图 1-2 所示），该图从全局角度阐明了决策管理带给组织的收益。

决策管理执行前与执行后的效益差量化了决策管理系统的投资回报率（ROI）。

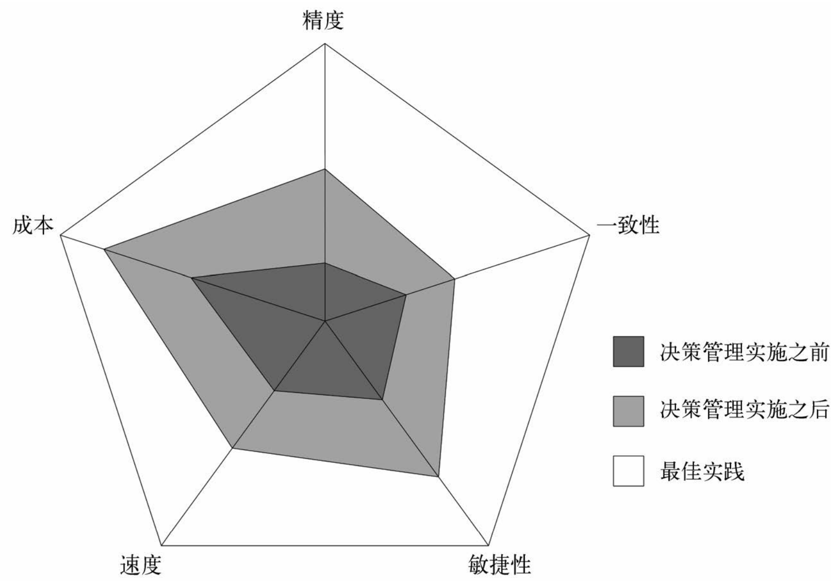


图 1-2　决策收益定价法

图 1-2 中使用的维度灵敏地概括了决策管理带来的核心收益。这些收益均来自于系统表述业务知识并使得业务决策得以自动化的能力。这一内容如此重要，因而此处再强调一次：决策管理带来的所有收益均源自于其能够系统地表述决策相关的业务知识，以及能将之封装于决策自动化系统中的能力。

决策收益：决策管理的收益
精度：在每个案例处理过程中都实现准确、可靠、理想的决策制定。
成本：在每个案例处理过程中都减少运营成本。
速度：处理案例过程中减少所耗时间。
敏捷性：允许业务用户快速、便捷地更新业务策略。
一致性：确保业务政策在所有渠道、产品和案例中的持续一致性。

知识创建与封装所用的工具和技术将在第 3 章进行具体描述，本章我们先简

要地概览一下如何运用它们来扩展决策收益定价法中的各个维度。

- **增加精度**　借助基于规则和分析模型等形式的知识，决策管理系统往往比传统决策流程做出的决策更加精确。若系统是基于规则的，那么它将模拟组织中最佳决策者的决策制定行为，并保证不会出现故意违法的情况。若系统是基于模型的，那么它将根据预期收益最大或其他重要业务指标最优的原则进行决策。无论基于何种执行方式，系统都会对每个单独业务采取相同的最优决策制定策略。通过持续监测自动决策的行为，可以确保系统中的知识总在不断更新并适应变化着的商业环境，同时使系统的性能并不会随时间推移而降低。
- **降低成本**　使用业务知识自动化决策制定流程而非仅仅使用行政管理的方法，将允许管理者更加灵活地分配人员力量。通过将大部分业务进行完全自动化处理，管理者可将人员集中在更加困难、更具风险甚至存在欺诈可能的业务上。它还可以帮助企业更加灵活地购买数据，例如判断是否购买机构数据，如若购买，还需判断哪些数据是有用的，以及从哪家机构能够买到这些数据，等等。
- **提高速度**　即便采用设计最为完善的决策支持系统，一个人类决策者也需要数秒甚至数分钟的时间才能做出一个基本的接受/拒绝判断，而在一些更复杂的情况下，如计算一项健康保险索赔业务收益的时候，可能要花费数小时的时间。但当这些业务知识被获取并自动化之后，同样的决策可能只需要几微秒的时间。只有在使用决策管理系统建立起端到端的全自动发放流程之后，零售金融产品（如信用卡）的在线申请处理才成为了可能。
- **提高敏捷性**　如果业务知识是存在于人类决策者的头脑中的，那么变更公司战略就至少需要数天甚至数周的时间，需进行包括传阅新的文件甚至是对人员进行再培训等工作。而使用基于现代决策管理工具搭建的自动决策系统，业务知识可直接由管理人员进行升级，避免了常见的 IT 研发和管理周期变更。例如，若一名信用风险管理人员想要变更贷款申请人的信用级

别标准，那么他仅需要更新一张数据表，在虚拟市场中测试一下新的策略，最后部署新策略并替换掉原有策略即可。全程不过几分钟。

- **提高一致性** 一旦业务知识被获取并被封装入自动化决策系统中，它就可在不同的产品、业务线、渠道和市场部门中被重复使用。这就保证了客户待遇的一致性，同时降低了实施的成本并简化了所需要的维护。相比对每类商品都设置一个特定流程来处理的方法，使用决策系统使得交叉销售产品和对产品记录的处理变得更为容易。

图 1-2 是一个决策收益定价法的图解，该图对决策的性能及其改进的可能性给出了一个多维度和定性的展示。若要将其转化为单一货币价值（如为一个业务案例提供具体的 ROI 数据支持），企业需首先基于核心绩效指标（Key Performance Indicators，KPI）对转化指标达成共识，以便将定性的维度量化为美元（或其他货币单位），之后便可对结果进行加总。

这对于精度和成本维度来说并不太难，因为这两个维度原本就是关键财务指标。速度和敏捷性被认为在一定程度上降低了机会成本（以敏捷性为例，若原本升级战略的花销是 X 美元/月，决策管理系统可使战略变更落实的时间由六个月缩短到一个月，那么在每次战略升级中都可以省下 5X 美元）。一致性维度的量化较为困难，因其回报往往是客户体验提升、品牌声誉提高等这类抽象收益。

如果此类回报要被包含在 ROI 结果中，公司需要用客户忠诚度或交叉销售机会等数据来估算其所带来的收益。

本书并非一本决策收益定价法的教程，想要了解更多有关的内容，请阅读 Taylor[13] 或 Larry Rosenberger、John Nash[14] 的相关著作。此外，还有一些其他的方法也同样可以有效审计决策管理项目所带来的收益。上述内容的主要目的是，阐明当业务知识被嵌入自动化决策系统之后，它将为公司带来一系列特定收益且这些收益可被量化为货币价值。

这些收益的价值正是知识的价值。

1.3　参考文献

1. R. M. Solow, "A Contribution to the Theory of Economic Growth," *Quarterly Journal of Economics* 70, no. 1 (1956)：65-94.
2. T. W. Swan, "Economic Growth and Capital Accumulation," *The Economic Record* 32 (November 1956)：334-361.
3. R. M. Solow, "Technical Change and the Aggregate Production Function," *Review of Economics and Statistics* 39 (1957)：312-320.
4. R. E. Lucas, Jr., "On the Mechanics of Economic Development," *Journal of Monetary Economics* 22 (January 1988)：3-42.
5. C. I. Jones and P. M. Romer, "The NewKaldor Facts：Ideas, Institutions, Population, and Human Capital," National Bureau of Economic Research Working Paper 15094 (2009).
6. P. M. Romer, Stanford University web page：www.stanford.edu/~promer/new_ bio.html (May 2011).
7. P. M. Romer. "Increasing Returns and Long-Run Growth," *Journal of Political Economy* 94 (October 1986)：1002-1037：and P. M. Romer, "Endogenous Technological Change," *Journal of Political Economy* 98. no. 5 (October 1990)：S71-S102.
8. Lucas, "On the Mechanics of Economic Development."
9. T. Debevoise, "Business Agility and a Meta-Knowledge Framework," blog posted November 7, 2010, www.tomdebevoise.com.
10. P. M. Romer, "Economic Growth," *The Concise Encyclopedia of Economics*, ed. David R. Henderson, (Liberty Fund, 2007).
11. J. Taylor, *Smart (Enough) Systems：How to Deliver Competitive Advantage by Automating Hidden Decisions* (Boston：Pearson Education Inc., 2007).
12. F. Rhode, "Little Decisions Add Up," *Harvard Business Review* (June 2005).
13. Taylor, *Smart (Enough) Systems*.
14. L.E. Rosenberger and J. Nash, *The Deciding Factor：The Power of Analytics to Make Every Decision a Winner* (San Francisco：Jossey-Bass, 2009).

第 2 章

业务流程中的决策

决策管理使用业务知识对决策制定流程实施自动化，从而实现业务知识的价值。因此，给人的第一感觉，一个决策管理项目可能包含三项工作：识别业务流程中的经营决策，收集决策所需知识，以及把知识封装嵌入自动化决策系统中。但决策管理并非如此简单。

原因之一就是：决策管理的目的是改善决策机制，而非简单地自动化当前业务流程。收集已有的业务规则仅可简单描述当前业务的处理流程。或许，它可以用于当前流程的效率分析或作为一个自动化项目的开端，但对于决策管理而言是远远不够的。

一个更为微妙的原因就在于：决策自动化不可避免地会改变该决策周围的业务流程。为评估决策管理的成本和收益，我们需要观察决策自动化前后的全过程。这表明决策管理和业务流程管理密切相关。此处有两个常见错误：(1) 在不考虑重新设计相关业务流程的情况下，试图实现自动制定决策的任务；(2) 未考虑决策自动化机制，便开始重新设计业务流程。流程重设计时，应始终将决策管理纳入考虑范围，并将其视为流程重设计的一种方式。

本章将重点讨论如何使用自动决策服务对业务流程进行建模及重设计，以使其替代并重构业务工作流的部分流程。

2.1　业务流程建模

业务流程管理（Business Process Management，BPM）是一门研究如何设计与实施业务流程以使其最大发挥组织（企业、公司、机构等）价值的专业学科。通常，一个业务流程可被视为组织为了实现某一特定目的而实施的一系列相互关联的活动。所有这些活动可由人（例如员工）执行，也可以由技术（比如计算机技术）执行，或由两者协同完成。

业务流程建模指定了一个流程的组成活动并阐明了各个活动之间的关联。流程模型的规范对于一个流程的合格性能（特别是流程在不同时间对不同参与人员显示出的可重用程度）至关重要。同时，它对于判断业务流程对组织产生的价值，以及判断人员或系统在业务流程中开展活动时对该价值的贡献，也具有十分重要的作用。模型通常表现为图表的形式。在模型图中，活动被连接在一起组成序列并形成流程流。许多图表及符号均可用于业务流程建模；本章中的示例主要使用由对象管理组织（Object Management Group，OMG）[1]定义的业务流程建模符号（Business Process Modeling Notations，BPMN）2.0。

流程建模通常包括将业务活动分解为不同层次的任务这样一个过程。流程顶层业务中的活动往往非常宽泛，可能用于描述整个部门或团队的功能。此类高层次的活动可被分解为由许多更小的活动组成的*子流程*，一直分解到原子活动的水平，所谓原子活动，指的是不可再进一步分解的单个活动。

一旦一个流程可在足够细致具体的粒度上建模，该流程便可由一个业务流程管理系统（Business Process Management Systems，BPMS）来实施。BPMS 即是这样一种用于控制流程模型中定义的活动序列的计算机系统：能够自动触发可被系统执行的活动，并可协调相关人员的参与（例如，为用户提供界面屏幕或向人们的工作队列发送作业）。BPMN 一个十分重要的特征为：BPMN *是可执行的*。我们可建立一个利用流程模型直接管理流程的 BPMS，并确保任何案例的执行活动序列都遵循模型规定的流程。

注意，BPMS 其实就是对活动序列的控制：几乎所有实际处理过程均由流程流调用的服务执行，而非流程流执行。这就是为什么 BPMS 有时被认为起“调度”

的作用：它协调所有服务并确保正确的数据于正确的时间在这些服务中传递。该观点对于理解如何在业务流程中管理决策机制是十分重要的。

业务流程建模

业务流程建模将业务流程具化为一系列可用流程图表示的活动。

BPMN（业务流程建模符号）是 OMG 定义的一项标准业务处理建模形式。

BPMS（业务建模管理系统）是一个计算机系统，它通过执行业务流程模型中定义的活动序列来管理业务流程。

2.2 决策点和决策服务

决策管理的内容包括将自动化决策机制引入业务流程，因此，我们可能需要一些方法来表示业务流程模型中的决策制定流程。为此 BPMN 2.0 使用了一种专门的“业务规则”任务（见图 2-1 的判断合格性活动）。该任务符号表示一个决策点：在业务流程中，该点自动调用业务知识并制定决策。业务规则任务是原子级别的（即不可再细分）且不包含用户交互。需注意的是，虽然业务知识可能在相当长的一段时间内都是有效的，但是使用这些知识的决策均是通过某一决策点利用当时的可用数据制定的。

当自动化经营决策时，最常用的方法是将决策所需的知识封装在决策服务中，并使用 BPMS 在业务流程中的决策点上调用这些决策服务。一项决策服务以一组描述案例的数据为输入，通过应用相关业务知识，返回针对该案例的一个或多个决策。这些决策结果可以随后被 BPMS 用于选择某案例在流程流图中的流通路径，也可以以数据的形式传递到其他服务中。

决策点和决策服务

决策点是业务流程中自动应用业务知识来制定决策的点。

决策服务以描述案例的一组数据为输入，并通过使用封装的业务知识，针对此案例制定一个或多个决策。

决策服务由 BPMS 在业务流程中的决策点上调用。通常，一个决策点仅执行一项决策服务，但一项决策服务可能会被多个决策点调用。

例如，图 2-1 展示了一个通过销售代理处理人寿保险申请的简单流程。从流程目的来看，它是面向用户的。第一项任务是从申请人处收集相关数据；第二项任务是在一个决策点上作出判断：判断合格性。为执行该项任务，BPMS 将申请人的相关数据发送给判断合格性决策服务并在该点接收判断决策（合格或不合格），同时提供制定该决策的理由。BPMS 根据决结果执行下一项任务。若申请不合格，则 BPMS 触发一项用户任务告知该申请者他/她被拒绝了，并附上决策服务返回的拒绝原因。若申请是合格的，则 BPMS 调用一个子流程完成保险的销售。

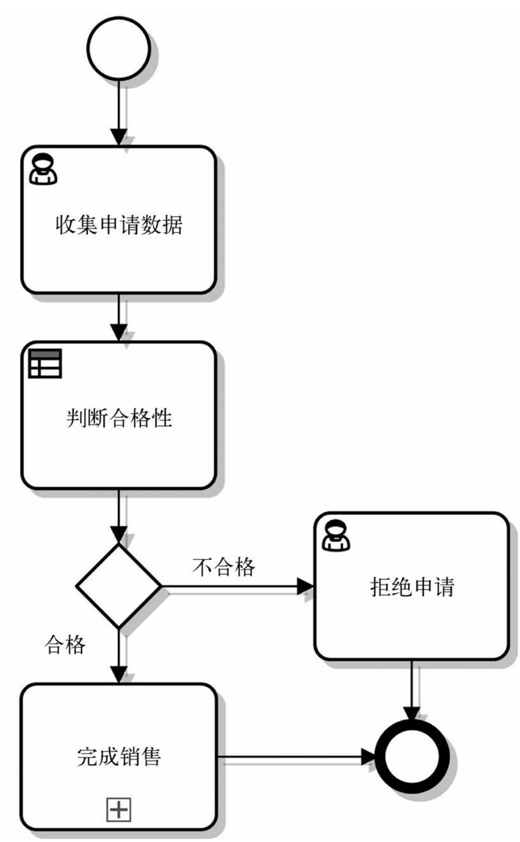


图 2-1　带有一个决策点的业务流程

一个更为一般的流程往往包含多个决策点。流程流中的每个决策点仅调用一项决策服务，但一项决策服务却可能会被多个决策点调用。即同一决策服务可被一个或多个流程中的多个决策点重用。这不仅是经济性的表现，而且也是一种跨越不同生产阶段、不同生产线、不同渠道、不同功能模块而保持决策机制一致性的可靠方式。

本书提到的决策服务均是*无状态的*，即决策服务没有记忆。当服务返回针对某一案例的决策后，该服务便会立即忘记与该案例相关的任何内容。若使用相同的数据再次调用该服务，它会制定*完全*相同的决策。而案例的状态——目前在流程流中的位置、收集到的所有数据及经历过的所有决策——或者保存在 BPMS 中，或者通过调用某项服务写入诸如数据库等此类安全的地方。创建一个“有状态的”决策服务不无可能，但这会引入许多技术上及逻辑上的难点，而这些难点并不在本书所讲方法的解决范围之内。

用于创建、部署以及执行决策服务的相关技术将在第 3 章中介绍。但从 BPMS 的角度来讲，任何执行特定决策服务的技术都是无关痛痒的。BPMS 使用的所有服务均是如此，重要的是服务的*接口签名*：接口的输入数据以及接口的返回结果。这种决策自动化的实现方法与面向服务的体系结构（SOA）方法的原则是一致的，同时该方法可清晰地将组织的业务知识与其他系统逻辑剥离开来。正如 Ian Graham 所描述的：

> 面向服务的体系结构及业务规则管理系统（BRMS）是现代敏捷企业不可或缺的一部分。它们大幅减少了由复杂且不稳定的商业战略和政策变化引发的相关问题。同时，SOA 和 BRMS 是两种平行互补的技术。
>
> 良好的 BRMS 可允许其应用程序被部署在一个 SOA 上。规则引擎因而可将自身作为应用的一个服务，且应用本身亦可被部署为服务（比如网络服务）[2]。

历经多年发展且拥有庞大遗留系统的企业会发现，其核心业务决策逻辑往往

贯穿于多个系统：用户界面系统、工作流管理系统、客户关系管理系统、会计系统以及数据库系统等。这会导致许多问题，如强耦合（若要修改一个系统逻辑则需同时修改其他系统）和模糊逻辑（某项可作为单条规则被实施的业务可能会被写成嵌套代码，并需要多个系统的参与配合方可实施）等。最近我一直在与英国一家经营多年的金融机构打交道，这家机构引入决策管理系统的最大障碍是现有的遗留系统太过复杂、混乱，没有人知道它是如何工作的。决策服务的引入为此提供了一种条理清晰且便于管理的方式。

2.3　业务流程决策的重设计

当为决策服务重新设计业务流程时，十分有必要将决策管理作为业务流程建模不可分割的一部分。已有流程中的决策机制往往需要许多人员在不同时间参与，而流程重设计则为整合该过程提供了一个独特的机会。一个良好的流程设计，其目标是定义一组高效、独立且可重用的决策服务。这些特点均可对如何使用重设计的流程中的决策服务产生影响。

决策服务设计目标

1. 高效性

- 将决策制定流程整合为少数决策服务。
- 仅为决策服务提供其必需数据。
- 杜绝使用相同的数据多次调用同一决策服务。

2. 独立性

- 在决策服务之间自然地划分决策制定流程。
- 最小化决策服务之间的交互。

3. 可重用性

- 一般化决策服务，使其可跨越功能、渠道及产品线被重复使用。
- 将知识封装在多个决策服务中，以实现知识的重用。

高效性：调用一个决策服务产生的大部分计算开销都在接口处——整理输入的数据，启动服务并返回结果——而非调用服务制定决策这部分。这表示：

- 应将一个决策制定流程整合为少数几项决策服务，而非将其分割部署在多个互不相关的服务中；
- 应仅将某一决策所需的数据提供给该决策服务；
- 不应使用相同的案例数据多次调用同一决策服务。

独立性：一个条理清晰的设计对于系统开发者和业务用户都十分重要，因为两者均需表达和维护封装在服务中的知识。业务知识具有自然结构，且其结构应该在业务流程的设计结构中有所体现。即：

- 通过封装连贯的业务知识，每项决策服务仅实施决策制定流程的一个步骤；
- 应尽量减少决策服务之间的交互，且交互仅限于使用之前的决策结果制定后续决策的情况。

可重用性：跨越服务功能、产品渠道以及业务线实现决策制定流程的连贯的一致性对于企业管理有众多好处。实现一致性最简单的方法就是共享常见的决策服务，或重复使用服务的决策组件。即：

- 决策服务应被一般化且可被重复使用，而非仅针对特定功能、渠道及业务线；
- 当整个决策服务不可被重用时，可允许对服务中封装的知识进行重复使用。

当我们着手处理业务流程中的决策自动化任务时，应一直牢记这些目标。任务可被分为三个主要阶段。

(1) 为组织的决策机制构建一个抽象模型。

(2) 重新分配员工在自动化决策服务中的角色。

(3) 合理化决策点及业务流程。

2.3.1 员工决策机制的建模

我们绝对不能低估任何组织中员工决策机制的复杂性。人为地处理单个案例

即可生成一张人际交互网，如案例会谈、建议请求、专家咨询、多级审批、监督管理及其他交互等。在经过复杂的人员活动和交互之后，大家会针对某一案例达成一组意见一致的决策，但人们往往并不了解其中的原因。如果打算重新设计如此复杂的决策制定流程，我们首先需要了解这个流程。

Paul Konnersman 提出了一种分析和建模员工决策机制的方法，能极大地简化这些看起来几乎不可能的任务[3]。他的决策流程规范方法基于这样的想法：既然许多决策会涉及多个人员，那么建模的“原子活动”应该是决策本身，而非任何特定参与人员的工作贡献。该方法认为每个决策都可被描述为一项由一个或多个角色的人员参与的活动。Konnersman 在其决策流程规范方法中使用了五种角色：每个活动有一个决策管理者，还可能包括咨询顾问、审批者、检查员和通讯员（如表 2-1）。

表2-1　决策流程规范中的决策角色

角　　色	责任对象	负责事宜
决策管理者：管理决策流程，制定决策，并负责实施决策	组织	提供一个及时、高效且有效的决策制定及实施流程
	咨询顾问	为其提供一个在决策制定之前改变决策的机会
	审批者	在决策制定后、实施前向其提交决策审批
	通讯员	在决策制定后及时向其提供决策的最新情况
咨询顾问：为制定一项良好的决策提供所需的专家意见；或为决策的成功实施提供所需的资源（可以影响决策却没有否决权）	组织	提供能够改善决策及其实施的专家意见或资源
	决策管理者	监督决策制定流程，为决策管理者提供相关专家意见，负责引导决策并为引导决策所造成的后果负责
审批者：避免出现由于决策者未参考专家意见而给出组织不可承受的后果的决策的情况；保护组织不会受到决策管理者的私人利益过度损害（具有否决权）	组织	确保决策管理者在制定决策时不会以牺牲组织利益为代价换取自身利益，从而迫使组织承担不可接受的风险
	决策管理者	在决策流程开始之前，提供尽可能清晰且详细的决策审批要求，及时向决策管理者提供决策审批通过或未通过的通知（附带审批未通过的原因）
检查员：确保决策结果符合规范要求（可拒绝检查）	组织	确保决策结果符合规范要求
	决策管理者	确保决策管理者在决策制定之前清楚决策结果的规范要求，并在决策制定之后尽快告知其决策的检查结果（包括决策未通过审查的理由）
通讯员：实施决策	组织	遵循一致性原则实施决策并执行所有后续任务以及制定所有后续决策

来源：P.M.Konnersman,“Decision Process Specification: A Methodology for Modeling and Enacting Professional Work Processes.”Working paper, 2006.

这些角色的交互规则可被编写为脚本的格式。每个决策均有一个内在流程，流程依赖于可能参与决策的角色而非决策本身。包含所有可能的参与者的抽象流程建模是一次性完成的，此处并不针对特定角色参与的具体决策流程进行建模。不同人员对于决策的贡献可被简单以角色（BPMN 2.0 中的“资源”）替代。此类抽象过程的简化可通过一个仅包含两个角色——决策管理者和审批者——的案例体现，如图 2-2 所示。

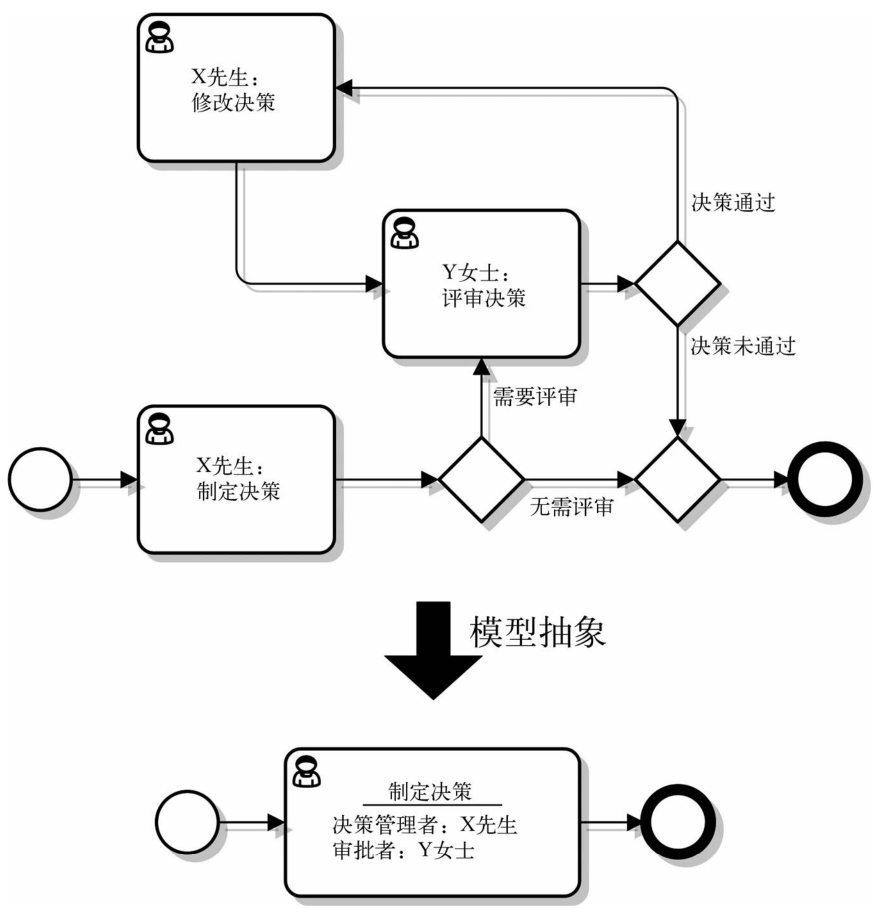


图 2-2　使用决策角色构建抽象模型

图 2-2 所示流程中，X 先生需对某案例制定一个决策。根据案例特点（例如，若案例的价值超过 1 万美元），该决策必须经由 Y 女士审批。若 Y 女士对 X 先生的

决策不予通过，X 先生就需修改决策直至 Y 女士审批通过为止。若将 X 先生视为决策管理者并将 Y 女士视为审批者，对其进行建模，该流程可被抽象为一项仅包含两个角色的活动。X 先生和 Y 女士之间的交互则被视为是该活动的内部过程。

涉及的交互角色越多，活动的内部过程就会越复杂，抽象模型所带来的简化效果也就越明显。如 Konnersman 所指出的：

> 若决策者和审批者均参与所有决策，传统方法就会错过将决策者和审批者之间的交互关系打包处理的机会。这种机会的重要性会随着决策参与角色数量的增加呈指数形式递增。[4]

抽象的结果是一个流程模型，在该模型中人员交互网络包含的每个决策——合作、审批、检查等——均可使用带有多重资源的单个活动表示。Konnersman 的目的首先是要提供一种流程建模方法，流程中所有活动均可被视为决策，其次是可在不考虑自动化能力的条件下，对决策建模。我在此处使用该方法的目的与其稍有不同。Konnersman 的分析方法对于流程重设计具有重要价值，可用于揭示已有决策制定流程的基础结构，因此可将其方法纳入一个新的、用于开发基于知识的服务的、自动化的流程中。

2.3.2　自动化决策制定流程

第 1 章介绍了决策收益定价法的概念并描述了决策管理在 5 个维度上能够带来的收益。精度、成本、速度、敏捷性和一致性，均可通过使用自动化决策机制替代员工决策机制得到改善。流程重设计的目标之一就是尽可能多的使用决策服务，而使员工仅负责与客户沟通、处理边缘案例和异常情况等。

现在普遍认为，这种“直通式的处理流程”能够使案例的处理时间及耗费精力最少（它既能降低成本又可提高速度）。但仍存在一种常见误解：这种方法会降低决策的质量，即它需折中考虑决策成本和决策质量。恰恰相反：采用决策管理进行决策自动化的同时提高了决策的精度和一致性。该方法同时提高了敏捷性，并可更加简单迅速地修改嵌入在决策中的业务政策。

然而，将决策服务嵌入主业务流程流极大地改变了人们在决策制定流程中所扮演的角色。管理变更通常是最后才被考虑的要素；流程已被重设计过，员工也已根据自身所需扮演的角色接受了再次培训——这几乎是流程重设计的副作用。但是从决策自动化的角度来看，员工角色的改变对于流程重设计至关重要；实际上它可被视为流程重设计目标的一部分。参与流程的员工的能力、所需的再次培训以及新角色所需的持续性支持都应作为项目前期流程重设计需考虑的内容。这不仅会对项目的解决方案施加限制，也是项目成功的关键。

变革管理的重心在于如何改变而非改变什么。例如，John Kotter 提出了一个包含 8 个实施步骤的方案：

(1) 营造一种紧迫感

(2) 产生引导性的合作

(3) 规划管理变革的愿景

(4) 传达该愿景

(5) 授权员工广泛的行为

(6) 生产短期回报

(7) 永不懈怠

(8) 将愿景融入组织文化中 [5]

许多转型努力的失败都可归结于缺乏愿景，但也有些失败案例表明细节决定成败：转型本身未被正确理解。从参与者的角度看待转型——设想参与者的角色将发生什么变化——在策划一连串有效的转型活动时非常关键。

在图 2-2 所示的带有审批流程的决策案例中，X 先生为决策管理者，Y 女士为审批者。当将该决策自动化并作为直通式处理流程的一部分后，一项决策服务替代了决策管理者，X 和 Y 的角色转变为仅负责处理棘手案例或异常情况的评审员。重新设计该流程的部分任务即为考虑如何重新分配 X 和 Y 在案例评审中的责任。一种选择是 X 和 Y 分别被授权两种不同级别的评审权限。表 2-2 可用于重新映射两者的角色。

表2-2　重新映射决策参与人员的角色

参与者	之前的角色	权限级别	现在的角色	权限级别
Y女士	审批者	超过1万美元	2级评审员	超过2万美元
X先生	决策管理者	不超过1万美元	1级评审员	1万~2万美元
决策服务	无	无	决策管理者	低于1万美元

两者的角色均在直通式处理流程中发生了改变：X 先生现在负责评审案例而非负责执行最初的决策制定；Y 女士的权限等级比之前更高了，因为她的部分评审工作现在可由 X 先生代为执行。理想情况下，我们应该对组织内的所有角色进行一个系统化的重新映射，这样当需要针对某种特定类型的决策自动化时，我们均可执行相同的映射角色策略，并使用一个前后一致的方法来升级参与者的职责。注意，在这个角色重映射活动中，决策服务与员工一起作为决策参与者并具有一定的权限。角色重映射的步骤如图 2-3 所示。

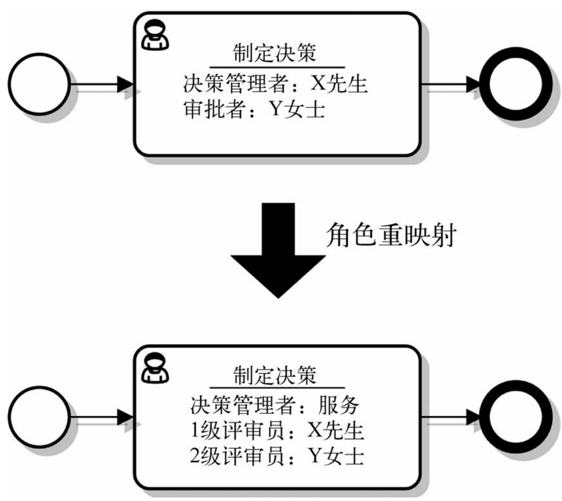


图 2-3　角色重映射

使用抽象方法为决策制定流程生成一个简单模型，并重新映射决策参与者的角色之后，我们即可扩展这些决策，并揭示整个业务流程自动化的完整模型。图 2-4 所示为之前所述简单例子的扩展模型：一项带有审批环节的员工决策现在可由一项带有员工评审环节的自动化决策代替。

在这样一个子流程中，一项决策服务（BPMN 2.0 将其建模为一项业务规则活

动）负责制定一个自动化的决策，同时负责对案例进行估值，并参考表 2-2 中的权限等级决定案例所需的评审级别。基于该决策服务返回的结果，BPMS 可无需任何人员参与直接通过该案例或者将其递交评审。这种简单的模式十分有效，并已被普遍用于自动化决策制定的直通式处理流程中。

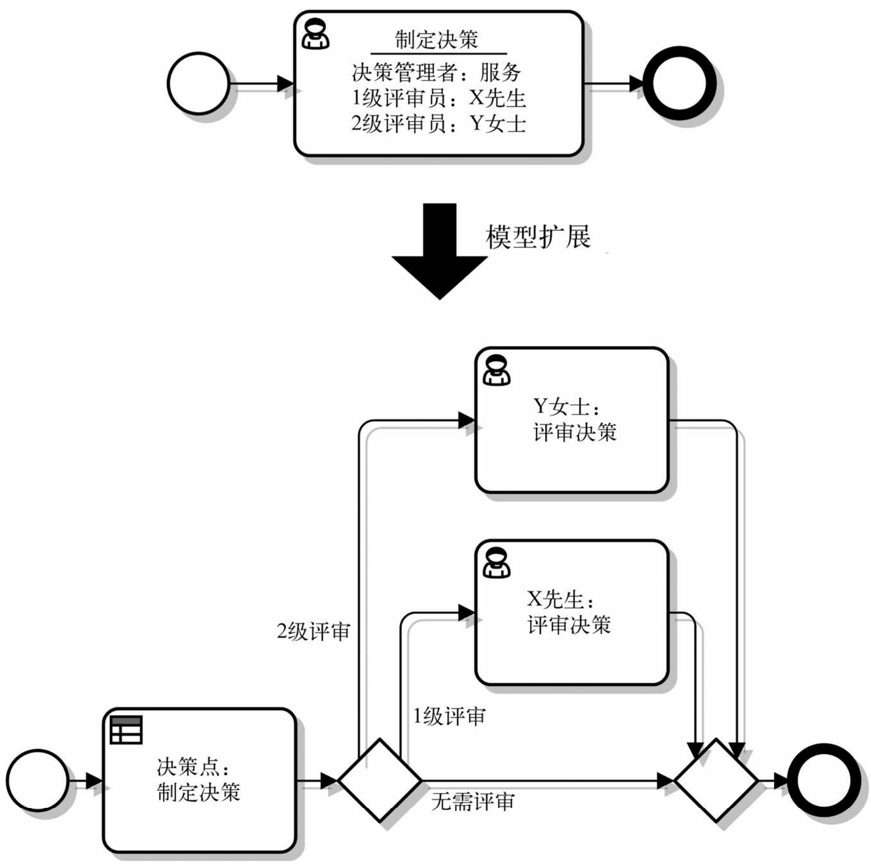


图 2-4 模型扩展

2.3.3 合理化决策点

上述示例着重强调决策的模型抽象、角色重映射以及模型扩展。但当我们对企业决策制定流程进行分析时，会发现它是一个由多项决策组成的网络。当将所有决策进行抽象、重映射及模型扩展后，其结果将会是一个包含多种决策模式（如

图 2-4 所示模式）的流程图，我们不应止步不前。业务流程建模是合理化及简化决策的一个绝佳机会，其目的在于实现本书之前所提出的目标：决策点的高效性、独立性以及可重用性。

合理化要求流程具有一定的灵活性。流程灵活性主要是指，一个决策可在两种情况下于流程中的任何决策点上被制定：(a) 点上的对应决策所需的所有数据均可从流程中获得，(b) 点上的对应决策返回的数据均为流程所需。这也使得将整个决策制定流程整合为少数决策点成为可能。

作为一个简单的例子，图 2-5 展示了一个包含两个连续“结合评审决策”模式

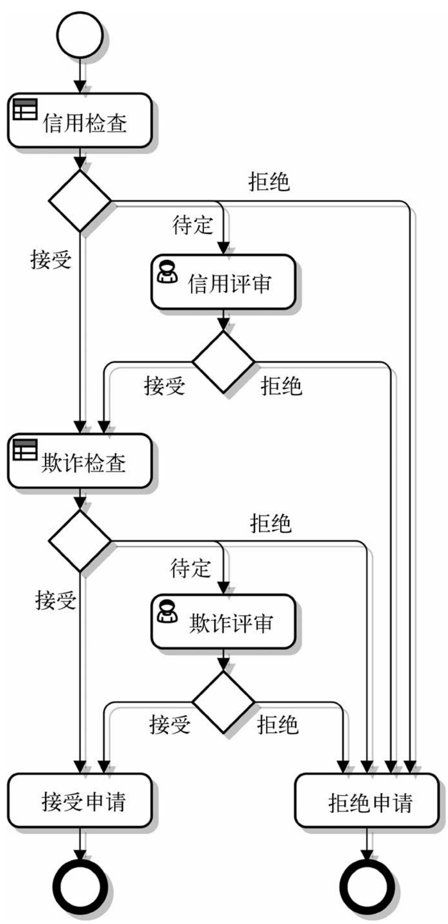


图 2-5　合理化之前：两个决策点

的流程：模式一用于检查和评审信用政策，模式二用于检查和评审欺诈可能性。每一种模式均使用一个决策点返回一项接受/待定/拒绝决策;这些结果将决定申请在流程图中的流通路径。这两个决策均基于之前收集好的数据，信用审查和欺诈审查这两个决策之间并未收集新的数据。这就意味着两个决策可在一个决策点上即信用及欺诈审查决策点被整合，如图 2-6 所示。

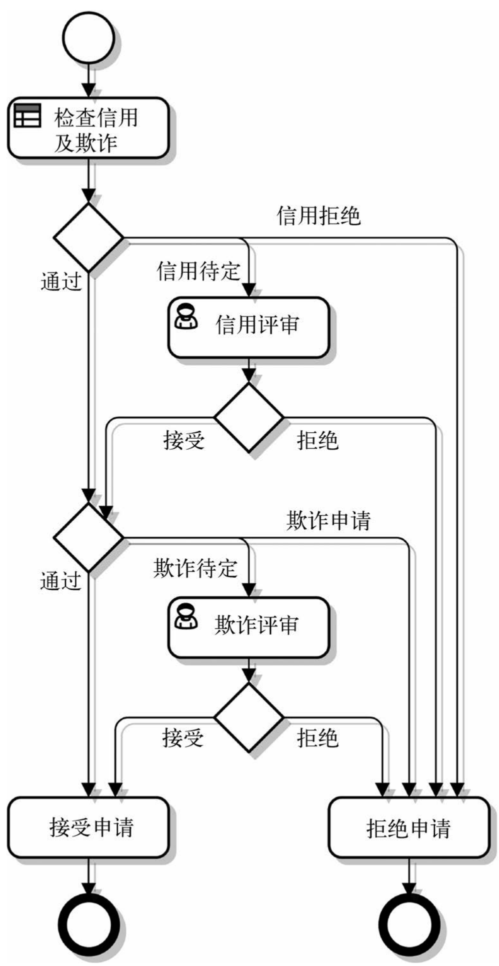


图 2-6　合理化之后：一个决策点

合理化决策点之后，通常情况下需重构并简化后续业务流程，但本书所提的敏捷性主要受流程的组织结构问题的约束。例如，图 2-6 中包含两个相互独立的分别

关于信用和欺诈的评审任务，且它们在流程中被顺序执行。这种情况浪费了流程处理时间，在该案例中，信用及欺诈审查可作为两个独立并行的处理任务，如图 2-7 所示。然而，这使得有时会出现已被信用评审节点拒绝的案例还需被欺诈评审节点处理的情况，因此这时我们在使用流程成本换取流程时间。最优方案为根据待定率和评审成本（各部门会有所不同，且这受到评审团队可用资源的约束）设计流程。

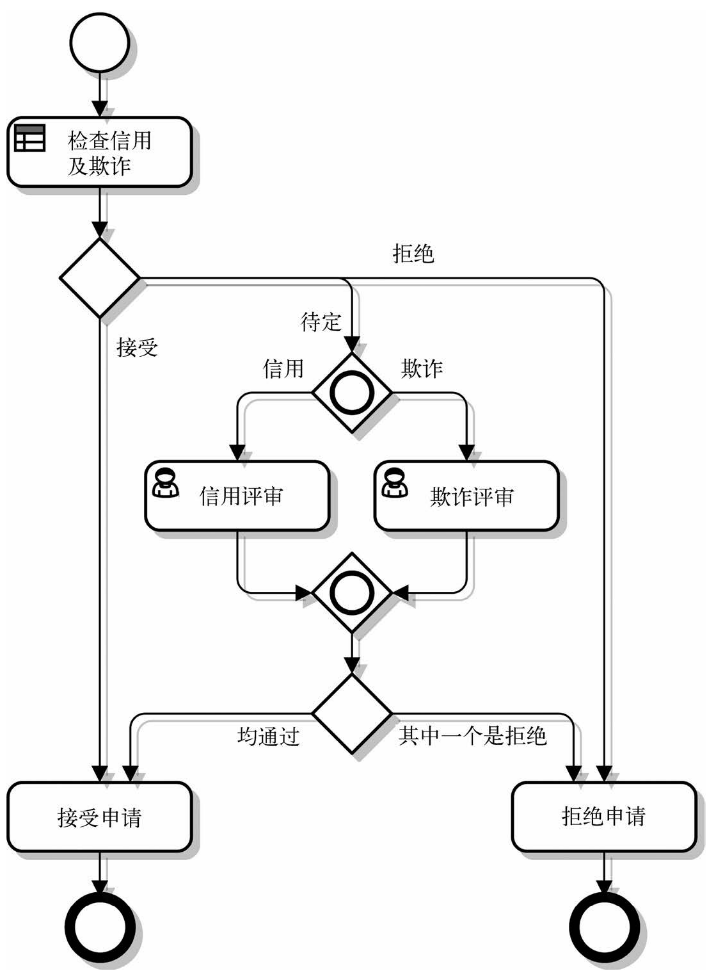


图 2-7　优化后续流程

注意图 2-6 的决策点会针对信用评审及欺诈评审返回两个独立的流通路径决策，但在图 2-7 中，它们被整合到一个单独的流通路径决策及一个待定列表（信

用或欺诈）。表 2-3 展示了决策重构的逻辑，它依赖于决策值之间的自然划分：“拒绝”优先于“待定”，而“待定”优先于“接受”。整合的流通路径决策决定了是接受、待定还是拒绝该申请；待定决策又会判断该案例是需被信用评估团队还是欺诈审查团队审查，还是同时需要被两者审查。这样的组合同时会提供一个拒绝决策列表，但图 2-7 中所示案例不存在这种情况。

表2-3　整合两个决策路径

信用流通路径	欺诈流通路径	整合流通路径	待定需评审	拒绝
接受	接受	接受	无	无
接受	待定	待定	欺诈	无
接受	拒绝	拒绝	无	欺诈
待定	接受	待定	信用	无
待定	待定	待定	信用，欺诈	无
待定	拒绝	拒绝	信用	欺诈
拒绝	接受	拒绝	无	信用
拒绝	待定	拒绝	欺诈	信用
拒绝	拒绝	拒绝	无	信用，欺诈

这一简单案例阐明了我们如何在不改变基本决策逻辑的前提下对决策进行重构。实际案例中存在许多可能出现的情况，如包含超过两个以上决策整合的复杂案例。该示例用于阐明两个原则：(1) 案例在业务流程中流通路径的业务逻辑应该尽可能地被包含在决策服务而非 BPMS 中；(2) 业务逻辑定义使用的语言应尽可能自然地反映流程流的结构。为阐明这些原则的作用，此处描述了一个带有决策点及路径选择逻辑的常用流程流。

知识自动化业务流程重新设计步骤

(1) 使用决策角色建模人为决策。

(2) 重映射自动化决策机制中的角色。

(3) 扩大决策范围，揭示自动化流程全模型。

(4) 决策点合理化。

(5) 优化后续业务流程。

2.4　发放流程模板

图 2-8 所示为一个关于零售金融产品（如信用卡、网上个人贷款等）销售的自动化发放流程的简单模板。该模板为低风险、高价值客户提供一个快速发放的直通式处理流程，它带有数据获取及员工评审的侧循环，且提供多条拒绝路径从

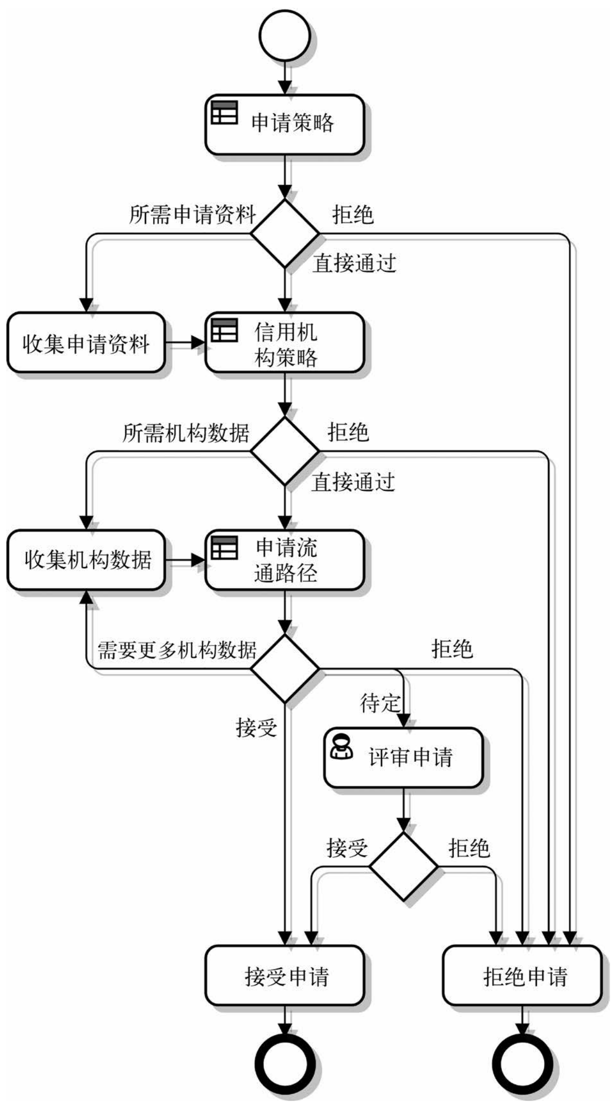


图 2-8　发放流程的主要决策点

而允许一旦确定某申请是不符合要求的，即可快速将申请移出流程。为结构清晰起见，数据收集及决策评审等此类活动均被描述为单个任务，尽管这些活动往往会更为复杂（如之前讨论过的）。在后续章节，当提到分析、制定及执行决策服务的方法时，本书将再次使用该示例流程。

这种类型的流程主要通过三个决策服务的使用实现自动化，本模板分别使用申请策略、信用机构策略以及申请流通路径为之命名。

申请策略决策服务将判定申请的类型以及申请所需的数据，举例如下。

- 若申请人已通过直接营销方式获得了预先批准，便无需再收集其他数据，可直接通过其申请。
- 若申请人是一个老客户且该客户正在违约中，他/她可能会被立刻拒绝。
- 若申请人是一个老客户且该客户正在申请一款新产品，我们只需了解他/她所申请的新产品信息而并不需要收集其个人信息。
- 若申请人是新客户，我们需要收集其全部相关信息，包括申请人的个人信息及其所申请的产品信息等。

收集申请数据任务可能会用于收集由某一渠道（例如，调用一组选定的用户界面屏幕）提供的申请策略决策服务所指定的数据。

信用机构策略决策服务使用收集到的申请数据进行判断：应该选择哪个信用调查机构以及从这些机构购买哪些所需的信用数据。该决策点的职能是以最低成本获取制定决策所需的信用相关数据，举例如下。

- 若已获得申请人相关的所有最新信用数据（这可能适用于所有的预先核准申请），则该申请即可在无需收集任何其他信用数据的情况下直接通过。
- 若申请数据足以表明该申请并不符合其所申请产品的发放规则，申请便可被拒绝。
- 其他情况下，决策服务会详细说明其所需要的信用数据（如信用记录，行为评分，欺诈指标）以及可用的信用调查机构的优先顺序等。

收集信用数据任务可能会用于收集由信用机构策略决策服务指定的信用数据，该任务会逐一调用指定的信用调查机构直至收集到所有需要的数据为止。

申请流通路径决策服务现已拥有判定申请流通路径及判定发放数额所需的所有数据。这项服务封装了公司的发放政策，如产品规格、信用风险以及偿还能力等，举例如下。

- 若已收集数据不足以支持一个明确的决策，且此时并未完全调用信用机构策略服务列举的信用调查机构，则可在收集更多数据之后再次调用申请流通路径决策服务。
- 若申请明显不满足产品发放的标准且产品没有降级销售的活动，即可拒绝该申请。
- 若申请满足产品发放标准或者产品可降级销售，申请即可被接受。
- 若申请的某些方面需要人为检查或判定，则将该案例送交评审。

申请评审是一项允许专家（如负责信用或欺诈评审的团队成员）评审申请业务并给出决策结果的用户任务。

在申请流通路径或评审团队给出决策意见之后，由接受申请任务或者拒绝申请任务完成整个发放流程。

2.5　流程设计方法

任何面向客户的自动化流程设计都存在以下三个层次之间的相互影响，以图 2-8 所示发放流程模板为例。

(1) 客户操作，包括流程申请、进展反馈、额度展示及最终决策通知等。

(2) 底层业务流程，包括数据收集、案例评审及案例终结等。

(3) 决策服务，包括引导案例进展以及允许决策机制自动化等。

Steve Towers 大力拥护的“由外至内”的方法着重强调客户体验的重要性，而传统的 BPM 方法则更关注于底层业务流程 [6]。但当采用决策管理自动化业务流程

时，这两个目标——为客户交互提供一个清晰的结构以及一组高效的业务流程——均依赖于决策服务的简洁设计（如前文所述，这是一种可实现高效性、独立性及可重用性的决策服务）。

后续章节将在 3 个更为细致的层次上对决策服务进行描述：

(4) 每个决策点上决策制定流程的结构。

(5) 每个决策所需的特定的支持知识（如业务规则等）。

(6) 以可执行软件的方式实施知识。

知识自动化层级
(1) 客户操作
(2) 业务流程
(3) 决策服务
(4) 决策制定流程结构
(5) 封装的业务知识
(6) 可执行的软件

该领域同样对于自顶向下及自底向上方法的各自优势有不同看法。James Taylor 和 Neil Raden 提出了一种自顶向下的决策管理方法，该方法在确定业务决策所需的业务知识之前，首先需对将要制定的决策进行定义[7]。而 Ronald Ross 及其他业务规则团队所追捧的业务规则方法则通过对业务如何得以执行施加通用约束，视业务规则为首要因素，并视决策服务为对业务规则的执行[8]。

然而，该领域也存在许多一致的观点，同时那些不同的观点大部分都只是侧重点不同而非实质上的分歧。因此 Ross 在比较了这两大阵营针锋相对的观点之后，给出了这样一个中庸的答案：

> 企业决策管理（Enterprise Decision Management，EDM）领域的理念在于：当你想表达这是一个“决策”时，不应该说这是一个“规则”，即

> 真正使业务增值的是那些使用规则制定的决策，而非规则本身。换言之，规则只是实现一些重要业务目标及制定经营决策的方法。因此，EDM 传达的思想为：当我们关注的是业务目标时，你不应该找企业管理人员探讨实现该目标的方法，这本身就是错误的。
>
> 这是一个很好的观点：并非规则本身使企业运作更为智能化，而是规则的使用可实现更好的决策，从而使企业运作更加智能化。规则及管理规则只是实现该目标的手段。我本人十分赞同上述观点并认为该解释十分合理[9]。

以上所述观点与本书所提的方法并不相互矛盾，我认为这是一种决策管理领域大部分从业者都会认同的方法。当自动化业务流程时，宽泛地讲该过程应该是自顶向下的，但在每个层次上，活动又会受到上一层业务需求及可能情况的引导及限制。

纯粹的由内至外的方法并不适用于顶层的三个层次设计。不考虑自身所提供的客户服务及实施的业务流程，而单独设计一组决策服务是不可能的。但纯粹的由外至内的方法也是不合适的。当设计面向客户的服务及底层业务流程时，应牢记必须使用一组合理且高效的决策服务来执行业务流程。因此，要实现决策管理的优势，必须要正确认识这三个层次的重要性并在流程设计中将其作为一个整体进行考虑。

同样，针对底层的三个层次，我将提出一种自顶向下的方法来分析每个决策点上决策制定流程的结构，发掘每个决策需要的知识并以可执行软件的方式将其实施。决策分析和知识发现必须受到某些规则的引导和约束。必须牢记：实施知识的可用技术必须能够使得知识的实施是高效且可维护的。我们真正需要的是一种“带有预见性的自顶向下”的方法。

因此，在描述本书的核心内容——决策分析及知识发现的方法论之前，本书将首先介绍在决策服务中封装业务知识所需的主要技术。

2.6　参考文献

1. Object Management Group, *Buiness Process Model and Notation (BPMN) Version 2.0*. www.omg.org/spec/BPMN/2.0 (2011).

2. I. Graham, *Business Rules Management and Service Oriented Architecture: A Pattern Language* (Chichester, UK: John Wiley & Sons, 2006).
3. P. M. Konnersman, "Decision Process Specification: A Process for Defining Professional and Managerial Work Processes. "*Portland International Conference on Management of Engineering and Technology*, July 27-31, 1997, Portland, Oregon.
4. P. M. Konnersman, *Decision Process Specification: A Methodology for Modeling and Enacting Professional Work Processes* (working paper, 2006).
5. J. P. Kotter, *Leading Change* (Boston: Harvard Business School Press, 1996).
6. S. Towers, *Outside–In: The Secret of the 21st Century Leading Companies* (BP Group Publishing, 2010) .
7. J. Taylor, *Smart (Enough) Systems: How to Deliver Competitive Advantage by Automating Hidden Decisions* (Boston: Pearson Education Inc., 2007).
8. Ronald G. Ross, *Principles of Business Rule Approach* (Boston: Addison-Wesley, 2003).
9. Ronald G. Ross, "A Case of Dueling Manifestos. Business Rules and Enterprise Decision Management," *Business Rules Journal* 8, no. 8(August 2007). www. BRCommunity.com/a2007/b357.html.

第 3 章

在决策服务中封装知识

人类的标志性特征之一即为语言，这是一种使用约定俗成的符号作为人类自身文化的一部分并用其进行交流的能力。自从我们的大脑进化到能够处理象征性姿势与言论，它便同时也具备了使用其他象征形式来传达和保存人类知识的能力。自此，人类文化的范围得到了进一步扩展，涵盖了史诗、歌谣、文本、图画、地图以及图表等。以文化形式流传下来的知识是具有进化价值的，它使得个体的成功经验可为他人所用，即便个体之间素未谋面。

尽管不再关乎生存或繁殖，知识依旧具有实用价值和经济价值，它赋予人类做事的能力。凭借一张地图，我就可以找到那些绘图者们曾经走过的、通往我从未到过地方的路线。严格地说，这种记录性的知识是仅仅介于两人之间的一种交流手段：一个人负责制定知识，而另一个人解读知识。但是，最近也出现了无需单方或双方参与的知识表示方法。

- 一些形式的知识无需由人类解读，即可由机器直接执行。例如，使用道路网中的机器可执行知识，GPS 导航系统可自动为你寻找抵达目的地的最优路线，并全程导航。
- 一些形式的知识可以由机器生成。GPS 导航系统使用的很多知识都是由系统从不同资源中半自动地编译和更新生成的。

业务流程中的决策自动化包括：识别并生成决策所需的业务知识，同时将业务知识编译成机器可执行的形式。随后知识可被封装于执行决策的决策服务中，

从而实现知识自动化。

本章简要介绍了业务流程自动化中用于构建决策服务的三项最常用的技术内容：业务规则、算法和预测分析模型。它们均可使用机器可执行知识的形式进行表示。业务规则和算法均为机器可直接执行的知识形式，但需要由人类专家构建。预测分析模型则更进一步：它允许机器使用历史数据自动生成新的机器可执行知识。

用于生成、部署和执行决策服务的这类软件工具通常被称为业务规则管理系统（Business Rules Management Systems，BRMS）或业务规则引擎（Business Rules Engine，BRE）。其实这两个术语都有一点误导性，因为（后文将介绍）业务规则通常只是整个服务解决方案的一部分，甚至是极小的一部分。一个有些过时但是准确得多的术语是“基于知识的系统”（Knowledge-Based Systems，KBS）。它涵盖了可进行业务知识建模并将知识部署于决策服务之中的任何系统。但是，我认为使用目前业界普遍应用且易于理解的术语则更为重要。所以，本书依然使用 BRMS 来表示生成和管理业务知识的系统，并使用 BRE 来表示 BRMS 中真正执行知识的组件或模块。

决策服务工具

BRMS 业务规则管理系统是一项可对用于业务知识进行建模并将业务知识部署为决策服务的软件工具。

BRE 业务规则引擎是 BRMS 中真正执行决策服务所封装的业务知识的组件。

BRMS 和 BRE 能够建模并执行多种形式的业务知识：不仅包括业务规则，还包括算法和预测分析建模。

3.1 业务规则

业务规则是一个逻辑语句，它使用一组条件判定一个结论。业务规则通常使用如下形式表示：“IF <条件> THEN <结论>”。以下是业务规则的一些例子，均出自于我在不同领域参加过的实际项目。

流程自动化中的许多业务规则常用于路径选择决策，如第 2 章所示的发放模板流程中的规则，举例如下。

- IF 申请人信用报告中的客户纠纷指示器显示为“Y”，THEN 申请的流通路径为“拒绝”。
- IF 申请人正在经历债务咨询且其贷款目的为负债整合，THEN 申请的流通路径为“待定”，且参考指标为“信用”。
- IF 保险资格在接受治疗日期之前或在入院日期之前被取消，THEN 索赔的流通路径为“拒绝”。
- IF 一项医疗索赔，但医生并未提交阐明医疗细节的索赔表的第二部分，THEN 索赔的流通路径为“搁置”。

但是业务规则也可用于许多其他类型的决策。以下为一些示例。

- IF 资产评估日期早于贷款申请日期超过 100 天，THEN 资产评估结果无效。
- IF 无担保贷款金额小于规定的无担保贷款金额上限 and 贷款价值比小于规定的贷款价值比的上限，THEN 抵押是安全的。
- IF 申请类型为“已评分”and 请求的产品包括个人贷款或信用卡 and 请求的产品不包含现金账户，THEN 交叉销售现金账户。
- IF 信用卡类型为“白金”and 该卡的信用额度低于 2000，THEN 销售行为降级为“普通”类型。
- IF 申请人为老客户 and 申请人近期信用数据有效 and 申请人最近的信用评分并未介于 595 与 629 之间，THEN 所需的信用机构服务级别为“迷你”。
- IF 申请人是新客户 and 申请人的就业情况显示其为合同工人或个体经营者 and 风险等级低于 3， THEN 保证金不能低于贷款数额的 15%。
- IF 理赔类型为怀孕保险 and 条件编码为 660 或 670，THEN 起保日期为入院日期减去 52 周。
- IF 医疗项目为 MRI 扫描 and 入住医院并未在医疗保险许可的 MRI 中心列表上，THEN 该项目不予赔付。

如上所示，规则是离散且具有明确独立性的语句，每一条规则表达可执行业务知识的一个原子活动。独立性，意味着每一条规则的意义都完全包含在自身之中，并不依赖于其与其他规则之间的关系。这在传统的“IF…THEN…”编程语言中是不成立的。因为“IF…THEN…”语句在编程语言中是顺序执行的，修改或移除其中任一语句都可能影响后续语句的执行结果。业务规则（在规则集中）可用顺序无关的方式被定义。这简化了表达、实现和维护业务知识的任务，并允许业务领域专家及业务分析师直接使用规则，而无需 IT 开发人员的参与。

此处只对业务规则进行简单的介绍。更多内容请参考 Ronald Ross 的 *Principles of the Business Rule Approach* [1]，Barbara von Halle 的 *Business Rules Applied* [2]，和 Ian Graham 的 *Business Rules Management and Service Oriented Architecture* [3]。

3.1.1 对象模型

业务规则有一个非常重要的特征，即它由对象模型、事实模型及领域本体等结构语言中的词汇写成。对象模型是一个非常贴切的术语，因为它为规则中提及的事物提供了一个模型：一个精心为特定业务知识领域设计的模型。所以，对于一组判断医疗健康保险索赔的业务规则，该模型需要包含诸如索赔、医疗项目、病患、医院及医生等对象。

许多关于对象之间自然关系的逻辑，被包含于本体论而非规则中。例如，一项索赔与某一名病患正在医院进行的一系列医疗项目有关。规则中的条件和结论涉及的数据项从来都不是简单的变量，而是对象一些属性，例如病患的年龄或进行医疗项目的日期。这简化了规则，并使其可以使用类似自然语言的形式表达，从而易被业务领域专家所理解。

对象模型可以继承：一个对象可以是另外一个对象的类型或子类。例如，一名会诊医师可被定义为是医生的一种类型。在这种情况下，一名会诊医师可继承医生的所有属性，并且能够以此为基础扩展自身的属性。这表示会诊医师可以使用医生的所有属性（如专业）进行描述，同时可使用会诊医师独有的属性（如所属医院和科室）进行描述。

对象模型中的对象，通常被更准确地描述为类，即它们表示一个特殊类型中的一组实体。类可以*实例化*：任何一个对象的实体都是对象所属类的*实例*。实例也会继承其“父对象”的所有属性。例如，我的猫“咪咪”就是猫这个对象的一个实例，而猫则是动物的一个子类，等等。

3.1.2　推理策略

可执行此类知识的系统被称为业务规则引擎（BRE）。BRE 对每条业务规则进行一次单独的逻辑推理：如果满足规则条件，那么其返回的规则结论为真。有些条件可根据输入数据的值进行直接判断，有些则依赖其他规则返回的结果。这意味着这些规则可以形成一个包含许多条规则链的逻辑网络。在这样的一个逻辑网络中，主要存在两种可行的推理策略：正向链式推理和逆向链式推理。商业中可用的大多数 BRE 都在一定程度上支持这两种推理策略：

- 在*正向链式推理*中，BRE 从初始的已知事实集合出发，不断地对规则进行判断，直到找到一条满足所有条件的规则为止。然后，BRE 触发该规则，并将其返回的结论加入到已知事实集中，作为下一步推理的依据。重复此步骤，直至无法得出更多的推理结果。这种策略也被称为“数据导向”的推理，因为推理过程是从最初的事实数据出发，通过对规则链进行逐步推导，最终得出结论。若已给定一组固定的初始数据集，并需要以此为基础执行所有可能的推理，此时，正向链式推理即为最佳推理策略。该策略在本书讨论的领域中十分常见。
- 在*逆向链式推理*中，BRE 首先假设一个命题成立。系统先找到一条以该命题作为结论的规则，然后判断该规则包含的所有条件是否均可被满足。若其中某些条件涉及其他规则的结论，则必须依次评估相应的规则，直到完成整条规则链的推理。这种策略也被称为“目标导向”推理，因为该过程从期望的目标出发，不断地寻找支持目标结论的数据。若数据必须根据具体需求进行获取（例如，通过向用户提出一个问题或者查询一个外部数据库），并且只对特定结论感兴趣，那么此时，使用逆向链式推理策略更加合

适。通常，逆向链式推理被用于诊断分析，在考虑下一个诊断之前，需首先对当前假设诊断进行调查和排除。

但是这些基本策略都颇为朴素，在一个包含上千条规则的知识库中进行应用时，就会变得非常低效。大多数 BRE 使用非常复杂的算法来提升大规模规则集上的推理速度。其中最为重要的一个算法是由 Charles Forgy 开发的 Rete。[4] 目前大多数 BRE 都支持 Rete 或其新版本 Rete II 和 Rete III。实际上，Rete 将所有规则预编译到一个可以表达规则之间依赖关系的网络中，然后基于该网络选择最佳调度对规则进行评估。给定当前已知条件，该网络可在每次有规则被触发时对自身进行调整，这使得只有被触发的规则才会被安排评估。

3.1.3 规则集和规则隐喻

另外一种提高推理效率的方法是将所有规则划分为一个个规则集（越小越好），并且每次仅对单个规则集进行评估。通过使用一个更高阶的规则集或一个程序流程图，我们可对"何时应用某一规则集"这一逻辑进行定义。规则集与决策服务制定的决策之间可能存在某种很强的联系：例如，每个规则集制定一个决策，并只包含那些支持该决策的规则。除提高效率之外这样做有其他好处。首先，它允许在规则集或决策层面——而不仅仅是在整个决策服务层面——重用知识。其次，对于业务领域专家们来说，它简化了规则的定义和维护。本书将在第 4 章和第 6 章着重解释如何在规则集与决策之间建立关联。

一旦使用规则集的形式组织规则，即可使用其他形式或隐喻，对规则进行表示，从而为业务用户提供方便。主要存在两种形式的规则隐喻：决策树和决策表。

业务规则

业务规则是业务知识的一个原子，以 If（条件）Then（结论）的形式表示。

业务规则的条件和结论参考**对象建模**中**对象**的属性。

在**正向链式推理**中，BRE 评估那些使用当前可用数据作为条件的规则，然后将评估规则得到的输出结论加入可用数据列表。

在**逆向链式推理**中，BRE 评估那些结论必需的规则，然后将所评估规则条件中的所有未知数据加入所需结论列表。

规则被整合为规则集，在制定其对应的决策时，一个规则集内的所有规则都会被涉及。

规则集可以使用决策树和决策表等隐喻形式表示。

1. 决策表

若某一规则集中的所有规则都满足：

(a) 规则中的条件均依赖于同一个小规模输入变量集

(b) 规则的结论均为同一输出变量

则该规则集可很方便地表示为表的形式。决策表为条件中的每个属性提供一列输入取值，并为结论中包含的每个属性集提供一列输出。表中的每一行表示一条业务规则。当规则不会用到某些条件属性时，其对应表格单元被置为空，或标记为“不可用”。

表格的形式十分便于业务用户理解，且易于检查规则是否已涵盖所有决策场景。在系统实际运行时，BRE 将每个决策表作为一个规则集，并将每一行作为一条规则进行评估。表 3-1 展示了一张决策表。该表使用两个条件——抵押物品和应急储备金，给出两个结论——信用级别和最大贷款价值比。

表3-1　决策表示例

条件		结论	
抵押物品	应急储备金	信用级别	最大贷款价值比
Y	＞30%	A	95%
Y	＞20%，≤30%	B	75%
Y	＞10%，≤20%	C	60%
Y	≤10%	拒绝	0%
N	＞25%	A	80%
N	≤25%	拒绝	0%

2. 决策树

当规则集中的规则共享一些条件，且这些规则均用于判定某单一属性的取值时，它们有时可被方便地表示成决策树的形式。一个决策树通过一系列测试对一个案例进行归类：树中的每个节点测试一个单独的属性，节点延伸而出的树枝则表示该属性的所有可能取值。末端节点（又称为树“叶”）表示所有可能的结论。决策树向相关规则的映射并不像决策表那样显而易见：树中的每条路径等同于一条规则，路径上的每个节点（除叶节点外）都是规则中的一个条件。

对于业务用户来说，决策树也是一种比较自然的规则表达形式，特别是对案例进行分类或细分的那些逻辑而言。图 3-1 展示了一棵简单的决策树（大多数决策树都会有更多的树枝）。这棵决策树决定发放贷款的额度区间。它等同于一个包含 10 条规则的规则集，每个叶节点对应一条规则。其中的一条规则如下：

- IF 申请类型为新 and 细分结果为高档 and 申请分数≤745 THEN 额度区间为 C。

体现这条规则的路径已在图中使用粗体实线标出。

注意，对于不同案例，需使用不同的测试并设定不同的测试次数：新客户的额度区间可根据细分结果和申请评分进行判断，而老客户的额度区间则仅需依据其行为评分进行判断，即时客户可通过判断其策略编码和行为评分为之选择相应的额度区间。所以，决策树是非对称的。

规则的对称属性决定了究竟该选择决策表还是决策树作为其规则隐喻。强对称性的规则（即每条规则都具有相同数目的条件，且条件基于相同的共享变量）可很自然地表示为决策表的形式；若将强非对称性的规则表示成决策表的形式，则会留下很多“不可用”的单元，所以此类规则应使用决策树表示。

决策表和决策树均要求规则是*完备的*且*互斥的*。完备意味着规则必须能够涵盖所有可能的决策场景；互斥意味着在任意场景中仅能有一条规则被触发。对于决策表来说，若其所有可能的列值组合都存在已被定义好的结论，则它是完备的；若对于任一场景都只有一条规则适用，则它是互斥的。对于决策树来说，如果对

于任一案例都存在唯一决策结果，则决策树是完备并且互斥的。

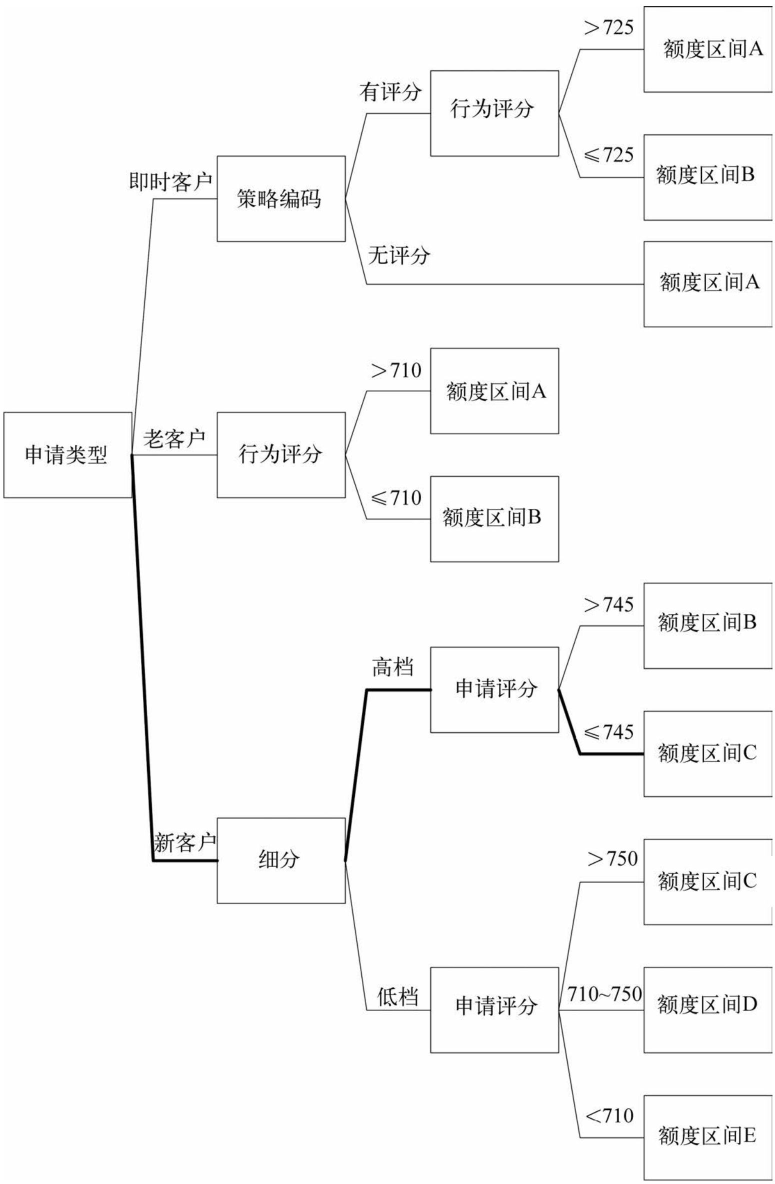


图 3-1　一棵决策树

不以决策表或决策树形式表示的规则集仍须具有完备性，但只要规则集是一致的，则无须具有互斥性。一致性表示：当一个案例同时触发了多条规则并产生

了不同结论时，存在一种可以鉴别出正确输出结果的机制。这种机制被称为“冲突消除”机制。这种特定的方法依赖于业务知识领域。例如，在一个路径选择规则集中，有些规则提议“拒绝申请”而有些规则提议“待定”。此时可能出现这种情况：一个案例同时触发了这两条规则，但此时我们要求给出一个唯一的决策。在许多场景中，可通过设定一组优先级别（例如，“拒绝”优先于“待定”，“待定”优先于“接受”）来解决这个问题，这种优先关系可被建立在本体论中。

规则集的特质

如果规则集能在所有可能场景中都能够做出一个决策，则它是**完备的**。

如果任意场景中只有一条规则能被触发，则规则集中的规则是**互斥的**。

满足以下条件时，规则集是**一致的**：(a) 规则是互斥的，或者(b) 当一个案例同时触发了多条规则并产生了不同结论时，存在一种可以鉴别出正确输出结果的**冲突机制**。

完备且互斥的规则集可以使用决策表或者决策树的形式来表示。

3.1.4 产生式规则及约束式规则

2002 年，业务规则组织（BRG）发布了一则公告[5]，某种意义上宣告了业务规则的独立地位。这份文件是一个言简意赅的关于业务规则使用原则的声明，它对于规范人们认识和讨论业务规则的方式产生了很大的影响。对于初次接触业务规则概念的读者，我建议将这份公告作为学习的入门教程。

尽管我发自内心地赞同公告中的大部分内容，但此处还是需要强调一下其中的两项原则。因为，乍看上去，这两项原则与我在本书中使用的方法并不是十分一致。

> 4.5 一条规则不同于为其定义的任何执行。规则与规则的执行是两个独立的问题。

4.6　对规则的定义应该独立于规则执行的人员、地点、事件及执行方式等具体内容。

3.1节一开始列举的4条决策路径的示例规则与这两条原则并不相符，因为示例规则中合并了业务政策与“如何执行业务政策”。BRG坚持认为业务规则应该是纯粹的陈述性约束，带有单独的“执行机制来表述规则被打破后会产生什么后果”。[6]我们用第一个例子来看一下，此规则将会产生什么影响：

- IF 申请人信用报告中的客户纠纷指示器显示为Y，THEN 申请的流通路径为“拒绝”。

以这种方式表现的规则被称为“产生式规则”，因为对它进行评估时会产生一个新的数据项（在这个案例中，新产生的数据项为申请流通路径的一个建议值）。若使用约束重写这条规则，即可写为：

- 在申请日期当天，申请人必须没有任何客户纠纷指示器显示为“Y”的机构报告。

此外，还需要有一个分离的执行规则表述：若在申请时打破了这条约束规则，则申请人应被拒绝。

根据我的经验，极少有情况需考虑业务约束中的抽象陈述。大多数实际情况中，业务规则仅在负责执行它们的决策点上被评估，并可能仅在这一刻两者是相关的。这也在我们的示例中成立：仅在考虑申请流通路径时会使用“拒绝正处于机构纠纷中的申请人”这一条业务政策。一旦产品已被销售，即使客户随后与某一机构产生了新的纠纷，产品也不会被取消。所以在这个案例中，没有必要为一条仅在申请时刻才会被执行的规则来建立一个通用的业务约束。这条约束规则可作为制定决策所需的知识的一部分。

该示例并不特殊：大部分规则都是或者仅在一个单独的决策点上被调用，或者在不同项目版本中的不同决策点上被调用。正因如此，本书使用的方法为：为

每个决策点单独定义其所需的决策规则，并将规则定义为同时包含使用约束及执行机制的产生式规则的形式。再次强调，这并不冲突，实际上，几乎所有用于业务流程自动化产业的规则都是使用这种方式定义的。

这种以决策为中心的方法生成的业务知识，并不如以抽象业务约束表示的业务知识灵活。例如，不能使用多种方式查询规则，且它们只能用于制定其所定义的决策，但该方法简单易行。通用业务政策约束所需的逻辑和本体论则相当复杂。例如，OMG 定义了一种标准业务词汇及业务规则的语义（Semantics of Business Vocabulary and Business Rules, SBVR），其内容长达 422 页[7]。具体的决策规则实施起来更加方便且执行起来更为高效，同时，我认为这也更易于对业务进行描述。

这并不是说泛化和重用规则就没有任何价值。我们的示例业务规则可能会在业务流程的两个决策点上接受检查，假设这两次检查之间隔了很长一段时间（例如，一个客户首先拿到了申请，然后 1 个月后才提供正式申贷额度）。如果是这样的情况，该规则就应该包含在一个通用规则集中，然后使用决策服务对其进行封装，并使用一个通用对象模型对其进行表示。这种通用规则集可对一些一般性的决策，如“某一申请是否合格”进行判断，而这一判断结果同时可被其他规则在对应的决策点判断申请流通路径。这在一定程度上区别了政策和政策执行。同时，我还注意到，甚至在 SBVR 的一个租车公司（EU-Rent）[8]的工作案例中，规则也被划分成了可在不同时间点上被使用的规则集，并且包含如“实际取车日期/时间”和“返还车辆日期/时间”这种表达方式的规则，这就将规则绑定在了特定的决策点上。所以，正如我们在第 2 章末尾所观察到的，也许这两种方法并非像一开始看起来那样差异巨大。

3.2 算法

在讨论决策管理时，经常会忽视那些“卑微”的算法，因为它并不像业务规则或预测分析模型这些新技术那样激动人心。但是，它却是封装业务知识所需技术中不可或缺的一部分内容，因为几乎每个决策服务都会或多或少涉及算法，而且有些服务会包含大量算法。

算法包含了用于计算某一结果的一系列指令。算法经常表示成伪代码（一种非正式但结构化的语言，类似于一种非特定的编程语言）的形式，但也可以使用流程图进行图形化表示。无论使用哪一种表现形式，算法中的基本成分都是相同的：进行操作的指令，如输入、计算和输出；以及操作指令周围的流程控制结构等（如“if…then…else”和“loop until”等将算法转到子过程的声明和调用）。

业务规则之所以如此引人注意，原因之一在于它具有良好的陈述性，这意味着，除了之前描述的那些优点，业务规则的定义还无需考虑顺序：由规则集制定的决策并不受集合中规则排列顺序的影响。但某些业务知识却与规则的执行顺序密切相关，因为某些特定的操作必须按照一定顺序执行。这对于计算而言，毫无疑问是正确的。业务决策经常会涉及大量财务计算，而程序或算法往往是表示这一过程的最佳选择。让我们来考虑这样 3 个层次的复杂性。

(1) *简单表达方式*。决策制定流程中用到的很多财务指标都是用比值表示的，例如，贷款价值比。这个比值可使用如下简单算式计算得出：

$$\text{LTV} = \frac{\text{贷款金额}}{\text{抵押物品价值}} \times 100$$

因此，它可被合理地囊括在一条简单的陈述规则中。

(2) *复杂表达方式*。更普遍的情况为：财务指标的计算需要综合考虑多个数据项。例如，要得到可支配收入，即需首先根据所有收入来源计算总收入，根据所有支出项计算总支出，然后使用总收入减去总支出，并对结果进行权乘。这可能比你期望包含在一条单一规则中的逻辑要复杂得多，但是它可被拆分为一系列规则。因为在此例中，数据项的计算顺序并不十分重要，它们之间的相互关系才更重要。

(3) *无表达方式*。有些指标根本无法被定义为表达式的方式（即无法以该指标作为一个等式的左边部分），但是可以使用一个复杂等式通过程序性的连续逼近方法对该指标进行估计。一个很好的示例便是有效年利率（EAR 或 APR）。

不同机构对于 APR 的标准定义可能会略有不同，但其共同的理论基础为：APR 是一种假设的利率，它可使得供贷方的预付费用等同于借贷方的还款（规费

及利息）费用。换句话说，它将供贷方所有的预付及手续费用打包成一个单独的贴现率。

基于如下等式，欧盟将 APR 计算方法规范化为一系列指令[9]，等号的一边表示等价的贷款，另一边表示还款的本金和利息金额（若你理解下面的等式，那么可以完全忽视此描述）。

$$\sum_{K=1}^{K=m}\frac{A_K}{(1+i)^{t_K}}=\sum_{K'=1}^{K'=m'}\frac{A'_{K'}}{(1+i)^{t_{K'}}}$$

其中：

K：借贷的期号。

K'：偿贷或付款的期号。

A_K：第 K 期借贷的金额。

$A'_{K'}$：第 K' 期偿贷的金额。

Σ：表示求和。

m：最后一期借贷的序号。

m'：最后一期偿贷的序号。

t_K：以年或年的等分表示的第 1 期借贷到第 2 期，以及后续每期借贷日期之间的时间间隔。

$t_{K'}$：以年或年的等分表示第 1 期偿贷到第 2 期，以及后续每期偿贷日期之间的时间间隔。

i：可通过（用代数方法或连续逼近法或电脑程序）计算得出的一个百分率，等式中的其他数据项均可从合同或其他文件中获取。

通常情况下，只有通过一个连续逼近的程序才能求得这个等式的解。首先估计 i 的一个值，使用合同中记录的借贷和偿贷步骤计算等式两边，检验两边的值是否足够接近。如果不是（第一步尝试时往往是这样），则在合适的方向上调整 i 的值，并再次计算。重复上述过程直到为 i 找到合适的值使等式成立。这就是 APR 的计算方法。

在某些 BRMS 中，可使用一组递归规则表示该逻辑，但这一方法并不十分直观。对于一个分析师而言，更自然的方式是使用一个程序表示这个连续逼近的过程。英国公平交易办公室使用这一方法解释业务应该如何遵守欧盟立法，在其发布的关于信用费用和 APR 的宣传册中[10]，提供了一种使用连续逼近方法（二分法）求解 i 值的算法。

```
        ; The Bisection Method(注释)
begin
        procedure obtain the amounts and times of the advances
        procedure obtain the amounts and times of the installments
        ; set minimum and maximum values for the range of i
        minimum_i=0
        maximum_i=1000
        repeat
        ;calculate i as the midpoint of the current range
        i = minimum_i + (maximum_i - minimum_i)/2
        procedure calculate 'total_PVs_for_advances'
        procedure calculate 'total_PVs_for_installments'
        ; find which end of the range to move to i
        if total_PVs_for_installment > total_PVs_for_advances
        then minimum_i = i
        else maximum_i = i
        end if
        until maximum_i - minimum_i < 0.0000001
        ; convert the result to a percentage
        print "APR =";  100 * i
        end
```

一般情况下，在决定使用业务规则还是算法时，最重要的考虑因素在于：哪一种方式更能够使主题事务专家更自然地表达其业务知识？

算法

一个**算法**包含了用于计算某一结果的一系列指令。

算法通常用于表示本身可被**程序化**的决策制定流程。

算法经常被写作**伪代码**的形式：一种非正式但结构化的语言，类似于一种简单的编程语言。

3.3 预测分析建模

既然这些决策都是由决策领域的专家参与制定，你也许会认为业务规则和算法可向你提供某个可能的决策业务逻辑的最佳描述。这是早期专家系统的宣言，当时的销售口号声称，公司的所有决策都应当与最优秀的专家制定出的决策一样好。但并非一定如此，因为还有许多专家不知道也无法知道的事情。Tom Davenport 使用下面一段话作为他的书 *Analytics at Work* 的开篇语。

> 如果我们想要做出更好的决策并采取正确的行动，那么我们必须利用分析学。分析学在实际工作中的应用即为使用数据和分析方法提高关键业务领域的绩效。长久以来，管理者依赖他们的直觉或“胆识”进行决策。长久以来，重要的决策都不是基于数据，而是基于决策者的经验和独立判断制定的。研究表明，40%的重大决策都不是根据事实得出的，而是根据管理者所谓的“胆识”。[11]

人的知识有时并不可靠，原因有两个方面。

(1) 它基于个体的经验，因此可能在整个业务领域中并不通用。

(2) 人类记忆并不完美，并且人的判断受大量的认知偏见的影响。[12]

预测分析讲述的是使用从数据中自动生成的人工业务知识代替或扩充人类自身累积的业务知识的内容，因为自动生成的业务知识不受上述两个问题的影响。通过分析历史数据，记录一组具有代表性的案例情景，然后使用分析结果来创建可客观预测类似案例发展的预测模型。

分析学是一个宏大而复杂的领域。其大部分内容关注如何为高级管理人员提供决策支持。正如第 1 章中讨论的，这是商务智能的领域，在本书中不做讨论。第 3 章内容介绍创建模型所用的主要技术，以及如何使用这些技术为经营决策部署决策服务。类似业务规则，在此我们只给出一个简短的介绍。参考书目请见 Larry Rosenberger, John Nash 和 Ann Graham 的 *The Deciding Factor*，[13] 以及 Tom Davenport，

Jeanne Harris 和 Robert Morison 的 *Analytics at Work*。[14]

预测分析

预测分析是一组从历史数据中自动生成业务知识的技术。

生成的知识以**分析模型**的形式展示，可以用来预测类似案例未来的发展趋势。分析建模技术包括**归纳**、**计分卡**以及**神经网络**。

3.3.1　规则和决策树归纳法

归纳法是根据对特定观测对象进行总结，从而得出一组一般性规则的一种推理方法。归纳法与演绎法形成鲜明对比，后者表示一种相反的过程：从一般性的规则出发进行逻辑推导。归纳法可以用来从案例数据中自动生成规则和决策树。

通过使用一个被称为归纳逻辑编程（Inductive Logic Programming，ILP）的过程，可生成一个能够制定单一逻辑决策的规则集。术语 ILP 由 Stephen Muggleton 最先提出。[15]若要使用 ILP，首先必须创建两组案例：一组正面案例和一组反面案例（例如，一组应该被拒绝的申请案例和一组不应该被拒绝的案例）。ILP 系统将这些数据解读为事实，并且使用逻辑归纳和概括方法生成一组与事实相符的规则。当输入一个新案例时，归纳生成的规则集即可判定是否要拒绝这个案例。

若提供的事实是完备的（涵盖了将被归纳规则集处理的所有可能案例）且一致的（对相似案例的决策不是相互矛盾的），则 ILP 可保证能够生成一个工作规则集。这在连贯、条理分明的领域，例如业务政策、规则、立法等领域是可实现的，这些领域也是 ILP 可以发挥的领域。然而，ILP 很少用于生成需使用人类行为因素（例如预测客户是否会拖欠贷款）作为条件的规则，因为人类行为具有不一致性。很有可能数据库中存在两个类似的客户，其中一个拖欠了贷款，而另一个却没有。最新研究正在致力于开发使 ILP 与概率推理相结合的方法，从而更方便地使用形如“IF x and y THEN 大概 z ”的自然归纳法。这种方法可处理人类行为数据，并做出真正的预测。但目前看来，人类行为数据处理最常用的是决策树学习

和评分卡方法。

决策树可通过对历史案例数据库的学习获得。每个案例都被描述为若干输入变量和一个目标变量的形式，后者正是我们希望使用决策树进行预测的。例如，如果有一个记录了每个客户年龄、性别、职业及收入（输入变量），以及他们实际上购买了什么产品（目标变量）的数据库，我们即可创建一棵决策树。该决策树在给定一个预期客户的完整输入时，可预测客户将会对哪些产品感兴趣。

学习是一个循环的过程。从一个完整的案例集开始，找到合适的测试方法，然后基于一个单独的输入变量，将案例集分成两个子集，并使子集内部目标变量的值尽可能相似，不同子集的目标变量的值尽可能不同。该测试即成为决策树的第一个节点。接下来，对每个子集重复同样的过程，并将相应的测试节点加入决策树中。不断用测试拆分这些子集，直到每个子集中的所有案例都在目标变量上具有相同的值，或者无法找到更好的测试来提升拆分结果为止。这些无法再拆分的子集便是叶子节点。

基于对最优拆分的不同定义，产生了许多不同的案例集拆分方法。一种十分重要的方法为 Ross Quinlan 开发的 ID3 方法 [16]，及其后续版本 C4.5[17] 和 C5.0。这类方法使用熵的度量方法来识别能够提供最大信息增益的测试。

一旦学习完成，作为结果的决策树即可以规则集（如前文所介绍）的形式在 BRMS 中执行。

归纳法

归纳是一种从具体观察中得出一般规则的推理形式。归纳能被用于自动生成规则集或决策树。

归纳法所得出的规则可以直接在 BRMS 中执行。

3.3.2 评分卡模型

Rosenberger 和 Nash 曾引述过他们 FICO 同事 Mort Schwarts 的一段话：

> 如果一个个人信贷员为狮子驯养员放出了 3 笔信贷但都失败了，那么他很可能不会再借贷给狮子驯养员了。但是，整个借贷组织对于成百上千名（而不仅仅是 3 名）狮子驯养员的投资经验可能大相径庭。人类大脑可以将这种定性反馈组织起来，以提高自身的决策能力。信用评分以及类似的基于计算机的分析工具的价值就在于，整个组织可以从大量数据中积累和加工投资经验，从而在企业层面上做出量化的目标决策。[18]

个人认为，尽管一个借贷组织不太可能有近千次借贷给狮子驯养员的经验，但上述观点依然很有道理。

评分卡是使用数值评分形式对某一特定事件的似然性进行预测的一类数学模型。例如，信用风险评分可预测客户拖欠贷款的似然性，欺诈评分可预测申请为欺诈贷款的似然性，破产评分可预测一个人在一定时间内申请破产的似然性。信用评分方法是 20 世纪 60 年代由费尔艾萨克公司（现在的 FICO 集团）创始人 Bill Fair 和 Earl Isaac 率先提出的。

要创建一张评分卡，必须首先有一个同时包含发生了某一事件的案例与未发生该事件案例的巨大数据库，通过对比这两组案例集建立评分模型。数据库中的每个案例记录了自身的若干特征（例如客户的年龄、性别、职业和收入），以及最终的输出结果（例如借贷已偿或拖欠）。统计技术常被用于检查某一特征在什么程度上可被用于预测输出结果。然后，最有效的预测特征（通常是 10 到 20 个）被组合和加权，从而获得更有效的有助于预测评分的案例特征。表 3-2 展示了一个包含少数特征的评分卡的例子。

表3-2　评分卡特征表

特　　征	取值区间	局部评分
年龄	18～21	10
	22～31	12
	32～41	14
	42～46	20
	47～56	22
	57+	25

（续）

特　征	取值区间	局部评分
性别	女	17
	男	15
	未知	16
职业	公司职员	12
	学生	23
	退休	26
	私营企业者	33
	其他	27
收入	<50 000	14
	50 000 ～ 74 999	18
	75 000 ～ 94 999	19
	95 000 ～ 149 990	21
	150 000 ～ 249 990	22
	250 000+	23

该示例评分卡蕴含了这样的业务知识：年长的申请人比年轻的申请人风险低，女性比男性风险低，退休人员和个体经营者比学生和雇员风险低，高收入者比低收入者风险低。该示例中的所有特征都仅包含单独的原始数据项，但是，包含由多项数据组合而成的复杂衍生特征（例如，家庭类型特征可通过申请人婚姻状况和家庭人口数量得出）的评分卡模型也十分常见。行业数据分析师的能力部分体现在能够找到从原始数据中获取预测特征的方法，正是这种技能使得手动制作的分析模型比完全自动从原始数据中生成的分析模型有效得多。

当在特定案例中应用此评分卡时，案例的每个特征都被用于查找相应的局部评分，然后，对所有局部评分求和。总评分表现了风险的排名，而不是一个概率值，其中高评分代表低风险。同时，评分通常被化为一个标准尺度（一般以 600 为中心值）。

注意，每个特征的贡献都独立于任何其他特征的贡献。所以，例如，不管其他特征如何变动，在这个模型中，年纪增大都会提高你的评分。评分卡无法对特征变量之间存在的相互作用进行建模。假若恰好年轻男性比年长男性信用风险高，但年轻女性比年长的女性风险低（当然，这是一个纯假设的例子），应如何建模？

一种办法是建立两个版本的评分卡，一个用于男性评分，一个用于女性评分。这种方法叫作*细分法*，该方法经常被用于需要细分案例以提高预测能力的评分模型。如此，计算一个评分需要 3 步：

(1) 为案例寻找合适的细分；

(2) 计算所有衍生特征；

(3) 选择合适的评分卡。

评分卡中特征的独立性意味着，知识可被很容易地表示成一组规则。表 3-2 评分卡中的前几条规则可表示如下：

- IF 年龄 >= 18 and 年龄 <= 21 THEN 得分加 10。
- IF 年龄 >= 22 and 年龄 <= 31 THEN 得分加 12。
- IF 年龄 >= 32 and 年龄 <= 41 THEN 得分加 14。

初始评分值为 0，当所有适用规则被触发后，评分将被校正并得出正确值。该规则集可在一个 BRMS 中执行。在某些 BRMS 中，评分卡作为一种标准形式的规则集隐喻，与决策树和决策表一并被提交与 BRMS。

评分卡有三大类。

(1) *申请型评分卡*，基于客户提供的信息，用于申请业务流程（例如申请表上的字段）的一个环节。申请型评分卡倾向于针对一条单独的业务线，所以企业通常使用自己的历史客户资料来开发在企业内部应用的评分卡。

(2) *行为型评分卡*，基于对客户使用特定账户行为的长期观察。例如，它可能会考虑客户的延迟还款历史记录，客户账户的收支平衡趋势，以及客户信用卡的消费金额和购买内容。

(3) *信用型评分卡*，基于信用调查机构提供的个人信用档案的内部信息。信用评分本质上来说也是一种行为评分，但是考虑了个人的所有账户及账户所属银行等内容。这种信用调查机构根据企业特定要求销售信用评分并将其作为一种服务。

企业有时通过合并多种类别的评分，以实现具有更强鲁棒性的决策：例如，一个内部的行为评分，一个内部的申请评分，以及多个机构的信用评分。

评分卡

评分卡是使用数值评分的形式对某一特定事件的似然性进行预测的一类数学模型。

评分卡总评分是通过对许多**特征**的**评分进行**累积求和得来的。

这些特征可以是单个**原始**数据项，也可以通过一组数据项衍生而来。

一张评分卡可以表现为一个能在 BRMS 中执行的规则集的形式。

申请型评分卡，基于客户提供的信息，用于申请业务流程的一个环节。

行为型评分卡，基于对客户使用特定账户行为的长期观察。

信用型评分卡，基于信用调查机构提供的个人信用档案的内部信息。

3.3.3 神经网络

在获得神经网络的博士学位后，我走出校园的第一次冒险是兜售我论文中的一个专利。有一家名为 Ferranti 的公司表现出浓厚的兴趣，这是当时一家十分优秀的全球性电子和计算机系统公司。但在 1981 年，神经网络只是一个研究课题而非一项被证实的技术，还存在许多亟待解决的重大问题，而 Ferranti 公司中并未有人计划对此进行投资。这项专利一直没有得到其管理层的关注，直到我和 Sebastian de Ferranti 本人共进午餐。他用 5 分钟的时间理解了专利的主要内容，然后给出了一个很直接的判断。“我认为，近期内这种技术将不会带来什么改变，”他说，“这一切都过于投机了。但看起来你是一个十分聪明的小伙儿，为什么不直接加入我们呢？”于是我成为了 Ferranti 公司的一员。

Sebastian 说得没错，至少对神经网络如此。事实上，我花了许多时间才使得该项专利走向成熟。但是现在，神经网络已是一项十分完善且有力的技术，并已被广泛应用于分析学及许多其他领域，包括语音识别、图像处理、自适应控制等。

神经网络形式多样，此处用于预测分析的原型为多层感知机（Multilayer Perceptron，MLP）技术，如图 3-2 所示。

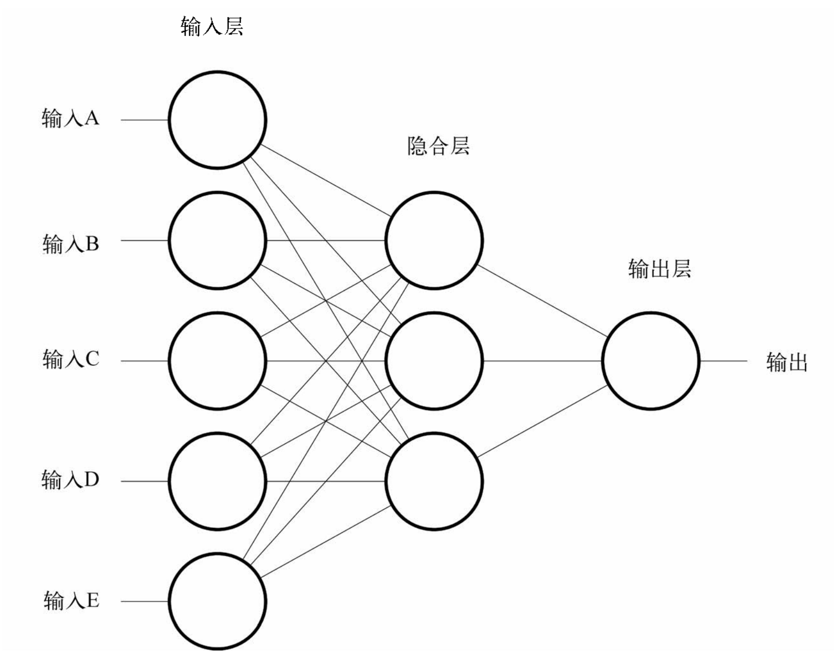


图 3-2　MLP 神经元网络

MLP 包含若干相互连接的节点。节点包含在层中——输入层、一个或多个隐含层，以及输出层——并且通常情况下，每一层的节点都与下一层的所有节点相连。每个连接都被赋予一个权重。当用于预测建模时，每个输入都代表案例的一个特征，而输出则代表一个评分，正如评分卡模型一样。

在 MLP 中，使用一组值表示案例特征，并将其置于输入层。然后，一波激活随后会从神经网络的输入端传到输出端。每个节点从它的前一层节点获得输入，将每个输入与连接上的权重相乘，并将所有的带权输入相加，然后将所得总和带入一个非线性方程中，之后将结果输出到下一层。输出层（图 3-2 只有一个输出节点）的结果即为总得分，而最初得分是随机初始化得到的。

神经网络可经过训练后做出有效的预测，训练方法为：将案例反复代入神经网络，并指定每个案例对应的正确输出结果。一种称为反向传播的算法可用于调

整权值。[19] 首先计算实际输出与正确输出的差值得到计算误差，在神经网络中反向传播误差，然后计算每个权重对误差的贡献大小，并以此为根据对连接权值进行调整。神经网络不断自我调整权值并使误差最小，直到其对所有的输入均可得到正确的输出。

神经网络的强大之源在于它的隐含层，这使得神经网络可以超越评分卡模型，学习非常复杂的非线性模型。因此，神经网络常被用于欺诈行为建模，它们可识别人类行为中常见的复杂欺诈模式。但是，该模型中的“隐含层”具有双重含义。

神经网络从未受到商业界的欢迎，因为神经网络学习到的知识仅可为自身所知。当建立一个评分卡模型时，任何一个业务人员都可解读它的特征并理解这个模型如何运作。但对于神经网络，即使训练好了一个模型，它的知识也只是表示为上百个数值化的权值，哪怕是最有经验的分析师通常也无法理解这种知识。所以，模型的置信度完全建立在神经网络在无数次测试中提供了准确预测的观察上。因此，一般只有在其他更加显性的建模方法不合适时，才会使用神经网络。

神经网络
神经网络是简化了的神经细胞的模拟网络。
MLP 包含若干层结点：输入层、一个或多个隐含层，以及输出层。
在神经网络中，学习的获得是通过节点权重的改变而实现的。
调整权重的方式之一就是反向传播误差。

3.4 汇总

规则、算法和分析模型这些方法并不是可以相互替代的，完备且有效的知识自动化需要综合使用这些方法。这三种形式的知识均有其各自的应用领域。

(1) 业务规则适用于没有历史数据可供归纳或者需使用业务政策和法律法规来定义知识的场景。规则在用于表示经常变换的政策（比如营销策略）时尤其适用，因为它们可以很轻松地根据业务专家的需求做出改变。

(2) 算法适用于遵循固定流程的计算。一般倾向于在静态的业务逻辑（例如，如何计算借贷的偿还金额）中使用算法，因为它们不易被频繁更新。

(3) 预测分析模型适用于需要以经验的方式获取知识的场景，特别是客户行为建模等。分析模型不常被更新，因为更新过程涉及收集新的行为数据和建立新的模型等大量工作。

何时使用知识汇总

业务规则适用于没有历史数据可供归纳或者需使用业务政策和法律法规定义知识的场景。

算法适用于计算遵循固定的程序化的步骤且步骤不经常改变的场景。

预测分析模型适用于需要以经验的方式获取知识的场景，特别是客户行为建模等。

自动化业务流程中的决策需使用 4 个主要的结构化组件，如图 3-3 所示。

(1) DWH（数据仓库）。数据仓库从 BPMS 和其他数据源（例如业务和客户关系管理系统）中收集数据，并使数据可用于报告和分析。

(2) PAM（预测分析建模）。预测分析建模环境允许分析师导入、过滤、操作客户数据样本，创建具有代表性的案例集以及使用这些案例来建立和测试预测分析模型。

(3) BRMS（业务规则管理系统）。业务规则管理系统允许系统架构师创建业务知识的存储仓库：包括导入的分析模型以及主题事务专家提供的规则和算法等，并将这些封装的知识部署在受 BPMS 控制的决策服务中。

(4) BPMS（业务流程管理系统）。业务流程管理系统通过调用决策点上绑定的决策服务并利用返回的决策结果对案例进行处理。所有的案例都被记录在 DWH 中。

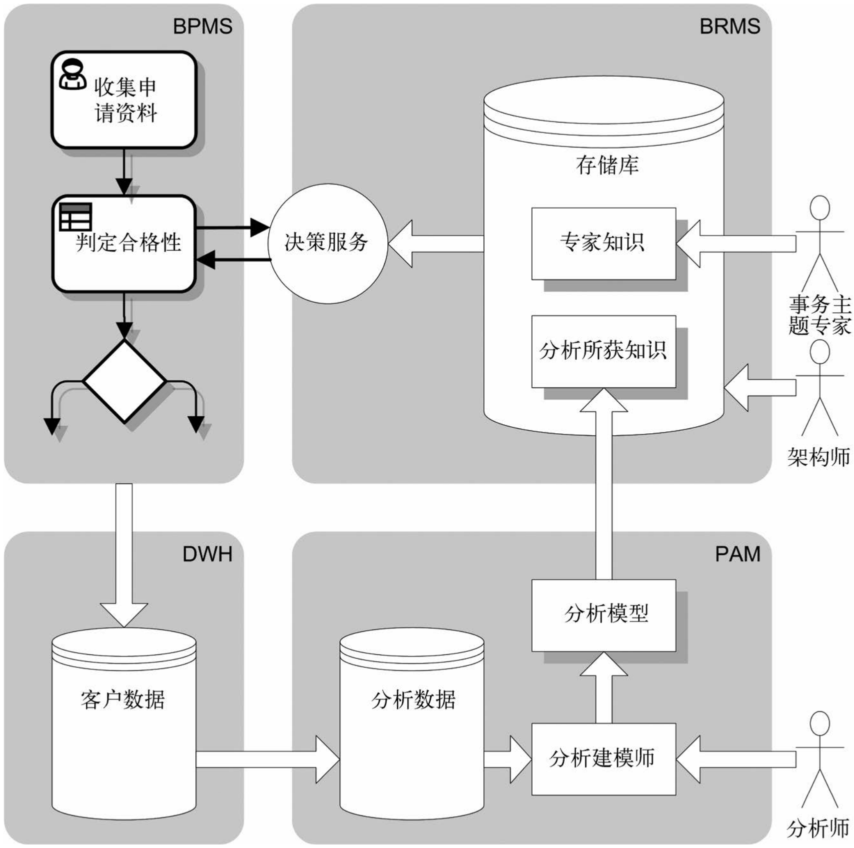


图 3-3　决策管理循环架构

注意，这 4 个组件形成了一个循环的处理流程。这个循环是决策管理的基本架构，因为所有业务知识都必须经常使用真实的决策对其进行测试。只有通过收集一段时间内的决策结果并对结果进行分析，才能验证业务规则的有效性（一种常用的方法是“冠军–挑战者”方法：使用两个或多个可相互替代的规则集随机地对不同的案例进行处理，然后使用统计的方法来比较其结果）。由于预测分析模型为客户行为建模，而客户行为随客户的经济状况和文化环境变化而变化，因此必须周期性地对其更新。

决策管理架构

DWH（数据仓库）收集并储存来源于 BPMS 和其他数据源的数据。

PAM（预测分析建模）环境允许分析师建立和测试预测分析模型。

BRMS（业务规则管理系统）将封装的知识部署在受 BPMS 控制的决策服务中。

BPMS（业务流程管理系统）通过调用决策服务来实施业务流程中的每个决策点上的决策。

该循环架构有时很难进行初始化，因为在一开始，组织可能极度缺乏建模所需的历史数据。但是存在许多方法可以跨越这个障碍。例如，业务政策最初可表示成一个业务规则集的形式，然后，随着数据的积累，可使用预测分析模型逐渐取代规则集。或者业务规则中可能包含一些基于权重或评分的条件，而这些评分和权重最初可由事务主题专家给出，随后可使用预测分析获得。

第一次使用这种决策管理架构是在 1996 年，当时我正在为英国税务局设计我的第一个规则系统。该系统用于评估个人所得税的回报。由于这在英国是一种相对较新的税种，英国税务局并没有大量的历史数据可供分析建模。于是，我在与一组高级税务稽查人员工作了几个月后，定义了近乎 500 个基于税务规则的特征。这些基于税务专家的专业技术和经验构建的特征，被认为具有很好的预测能力。隶属于这些特征的权重最初主要是基于税务专家们的经验设定的。与此同时，税务局启动了一个数据挖掘项目，可在数据积累的同时逐年调整这些特征权值。然而，最初的判断被证明具有惊人的预测性能。第一次运行的时候，我们通过一次批处理（使用了一整天的时间）向服务系统输入了英国所有个人所得税纳税人的信息。当看到排序结果时，一个稽查人员指着最上面的那个名字说道："我知道那个家伙，我关注他很久了。"

尽管图 3-3 展示了决策管理的基本架构组件，本书的重点并非是这种架构。本书主要强调如何构建并运行一个成功的知识自动化项目。任何项目的第一个关键步骤都是确定需求，但是知识自动化项目的需求非常复杂，可能包含许多由大

量知识元素组成的决策服务，以及成百上千条业务规则。这些规则有些是从人类专家中收集的，有些则由分析模型创建。如何界定、估计并规划这样一个项目？图 3-3 中箭头的方向为成功的项目自动化实施方法指出了一条途径：

- ❑ 业务流程需要决策服务
- ❑ 决策服务需要业务知识
- ❑ 业务知识需要分析模型
- ❑ 分析模型需要数据

而且，因为这是一个循环，最终还有：

- ❑ 数据的收集需要一个业务流程

但是，我们必须找到一个起点。本书的方法开始于业务流程以及为决策服务确定需求。我已经在第 2 章讲述了业务流程的来龙去脉。本章的内容则更进一步，介绍了需要包含在决策服务中的各种知识类型。第 4 章将会介绍决策需求分析（DRA）：一种规范化的、自顶向下的方法，用于确定决策服务所需执行决策的制定流程的结构，并可识别这些决策所需的知识和数据。第 5 章将会展示如何使用 DRA 的结果来确定项目范围、评估、规划和管理一个成功的知识自动化项目。

3.5 参考文献

1. R. G. Ross, *Principles of Business Rule Approach* (Boston: Addison-Wesley Professional, 2003).
2. B. Von Halle, *Business Rules Applied: Building Better Systems Using the Business Rule Approach* (New York: John Wiley & Sons, 2001).
3. I. Graham, *Business Rules Management and Service Oriented Architecture: A Pattern Language* (Chichester, UK: John Wiley & Sons, 2006).
4. C. L. Forgy, "RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Matching Problem," *Artificial Intelligence* 19 (1982): 17-37.
5. R. G. Ross, ed., *Business Rules Manifesto: The Principles of Rule Independence*. Version 2.0, Nov. 1, 2003. www.businessrulesgroup.org/- brmanifesto.htm
6. www.businessrulesgroup.org/defnbrg.shtml.
7. OMG. 2008. *Semantics of Business Vocabulary and Business Rules (SBVR)*, 1.0.www.omg.org/spec/SBVR/1.0/PDF.

8. Id.
9. European Parliament and Council of the European Union. “Directive 98/7/EC of the European Parliament and of the Council of 16 February 1998 amending Directive 87/102/EEC for the approximation of the laws, regulations and administrative provisions of the Member States concerning consumer credit.” *Official Journal of the European* Communities, L 101/17 (April,1, 1998).
10. Office of Fair Trading. *Credit Charges and APR: How to Calculate the Total Charge for Credit and the Annual Percentage Rate of Charge* (OFT144), 2007.
11. T. A. Davenport, J. G. Harris, and R. Morison, *Analytics at Work: Smarter Decisions, Better Results* (Boston: Harvard Business School, 2010). Citing Accenture survey of 254 U.S. managers; “Most Companies Say Business Analytics Still Future Goal, Not Present Reality,” Accenture press release, December 11, 2008, http://newsroom. accenture.com/article_disply.cfm?article_id=4777.
12. T. Gilovich, D. W. Griffin, and D. Kahneman, *Heuristics and Biases: The Psychology of Intuitive Judgment*. (New York: Cambridge University Press, 2002).
13. L. Rosenberger and J. Nash, *The Deciding Factor: The Power of Analytics to Make Every Decision a Winner* (San Francisco: Jossey-Bass, 2009).
14. Davenport, Harris, and Morison, *Analytics at Work*.
15. S. Muggleton, “Inductive Logic Programming,” *New Generation Computing* 8, no. 4 (1991): 295-318.
16. J. R. Quinlan, “Induction of Decision Trees,” *Machine Learning* 1, no. 1 (1986): 81- 106.
17. J. R. Quinlan, *C4.5: Programs for Machine Learning* (San Francisco: Morgan Kaufmann Publishers, 1993).
18. Rosenberger and Nash, *The Deciding Factor*.
19. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Internal Representations by Error Propagation,” in David E. Rumelhart, James L. McClelland, and the PDP research group (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 1: Foundations*. (Cambridge, MA: MIT Press, 1986).

第 4 章

决策需求分析

我从事知识系统构建工作已有 30 余年，根据我的经验，这类项目中最难的问题一般不是技术性的，这类问题很容易被解决。难点往往与管理或通信相关，例如：

- 如何在考虑了所有因素的情况下，在一个较高的层次上讨论所需明确的决策制定流程？
- 如何在尚未收集任何规则时，即在项目最初阶段清晰地定义项目范围？
- 如何在无需大量返工的前提下逐阶段实施项目？
- 如何避免项目开发过程中发生糟糕的意外情况（如整合一个额外数据源的需求）？
- 如何在多个团队或者地点间分配开发任务，并保证项目模块能被妥善集成？
- 如何确定我们已经完成了整个项目？

为应对上述问题，我设计开发了决策需求分析（Decision Requirements Analysis，DRA）这一方法。这是一种形式化的方法，主要内容如下。

- 决策需求图表（Decision Requirements Diagram，DRD）：该图表将所需制定的决策流程的结构表示为一个带有业务知识和数据支持域的、由决策或子决策构成的网络。

- 决策需求分析研讨会（Decision Requirements Analysis Workshop，DRAW）：一种结构化的研讨会方法，可以使用系统的自顶向下的方法分析决策机制，并将其结构记录在 DRD 及其支持文档中。
- 关于如何使用 DRA 进行项目规划、规则发现、功能设计和系统开发的最优实践指南。

第 5 章将详细讲述如何在项目管理中应用 DRA。本章将着重描述 DRD 和 DRAW，首先对一些基本原则进行声明。

4.1　基本原则

DRA 的基本原则非常简单。

(1) *仅传递需求*。任何项目的成功实施均依赖于两点：能够覆盖确定的项目范围，能够传递约定的需求。IT 业界已对项目管理的基本原则有了很好的理解。不幸的是，业务规则项目往往因其“范围蔓延”特性而声名狼藉，这通常是由于需求不明确以及缺少清晰的最终目标而导致的。

我们绝对不能因为业务规则项目很难就将其当作例外。尽管业务规则项目中实施的业务逻辑会非常复杂，但明确项目的功能性需求并与客户达成需求共识则往往更为重要。这与迭代开发的理念并不矛盾，我将在后面章节阐明这点。若在项目初始阶段就错误理解了项目范围，将会给供应商和客户带来灾难性的后果。

(2) *永远要自顶向下，而非自底向上地工作*。人们倾向于认为，业务规则可从组织中随意获取，然后被组装成有用的规则库。这种方法存在许多问题，主要问题如下。

- 一个组织可能有成千上万条业务规则，一个决策服务可能只需要其中很小的一部分规则来表示。
- 在不同的决策上下文环境中使用同一个独立的逻辑事实时，可能需要将其表示为许多不同的规则。

此外，仅仅收集和建模已有的业务规则属于人类学而非决策管理的研究范畴。我们的目标是改变组织中的决策制定流程，其唯一的可行途径是自顶向下的：定义所希望的业务决策，然后设计并开发相应的用于实施决策的决策服务。

(3) *决策服务制定决策*。该说明可能太过显而易见了，但令人惊讶的是，这个基本的事实经常被人忽视。原则上，决策服务是决策制定流程的一个基础组件。因此，尽管决策服务可被指定为多种方式（例如，一个函数，一个对象，或者一个从输入到输出的映射），最好的选择是单纯地根据它所要制定的决策来定义其功能。每个决策都是一种为某个问题提供一个答案的方法。所以，若你能清晰地定义问题、答案及其实现方法，你就完整地定义了一个业务规则服务的功能。

(4) *决策需要信息*。可以通过询问“制定一个决策需要哪些信息”来发现一个决策的需求。此处所指信息主要包括如下几种。

- 业务知识（可以是任意形式：业务规则及其隐喻，算法，分析模型等）
- 决策面向的案例的描述数据
- 其他决策的结果

其中最后一点十分关键：决策依赖于子决策。这就使得决策制定流程可被分解为一个由决策需求图表（DRD）刻画的网络。

决策需求分析原则
1. 仅传递需求
2. 永远自顶向下，而非自底向上的工作
3. 决策服务制定决策
4. 决策需要信息

4.2 决策需求图表

相比于文字，人们更擅长从图中获取信息，这也是图表普遍应用于每个专业

领域的原因。IT 行业更是依赖图表：仅统一模型语言（Unified Modeling Language, UML）2.3 就包含 14 种图表类型。[1]

不幸的是，我至今尚未发现其中任何一种图表被广泛应用于定义决策机制的需求。用例表确实是为需求定义而准备的，但他们将系统视为一个黑盒子而并不显示其内部的决策制定流程结构。实体关系图可用于获取知识结构（如 Barb von Hallen 和 Larry Goldberg 在他们的决策模型中展示的那样[2]），但相比于高层次的需求，它更适于对特定规则建模。

本部分内容将着重描述决策需求图表（DRD），它通常被用于项目的最初阶段（尚未发现任何业务规则时），可用于获取所需的决策制定流程的范围和结构。DRD 及其关联文档是 DRA 方法的核心。

4.2.1　定义

DRD 是一个网络图，它展示了在制定一个或多个决策时所需要的信息。从技术角度讲，这是一个有向无环图，即 DRD 是一个由许多单向箭头连接的节点组成的网络图，图中不包含环。它可以将一个决策制定流程分解为一组相互关联的决策及其支持信息域。图 4-1 展示了一个使用 DRD 进行决策的简单例子：如何处理一个抵押贷款的申请。与真实情况相比，该案例简单得不切实际，但却包含了一个 DRD 的所有基本特征。

该 DRD 包含三种类型的节点：决策、知识和数据。它们由表示需求关系的箭头连接：由 A 指向 B 的箭头表示 A 被 B 需要。举例来说，“风险可接受”决策的制定需要三方面信息：“申请风险评分”的决策结果，以表格形式展示的业务知识，以及信用机构数据。按照惯例，节点按照使箭头向上指的规则进行排布。注意 DRD 图表并不一定必须是一棵树：一条简单信息（例如，“申请风险评分”或“需要的贷款数据”）可被多个决策使用。

一个 DRD 仅由决策节点、知识节点、数据节点和箭头组成。

决策节点表示所需制定的决策。决策可以但并非一定由一项决策服务给出：决策亦可由一个人类决策者或其他一些系统组件制定。一般情况下，决策需求明

确之后即可将决策分配予不同的决策制定代理。

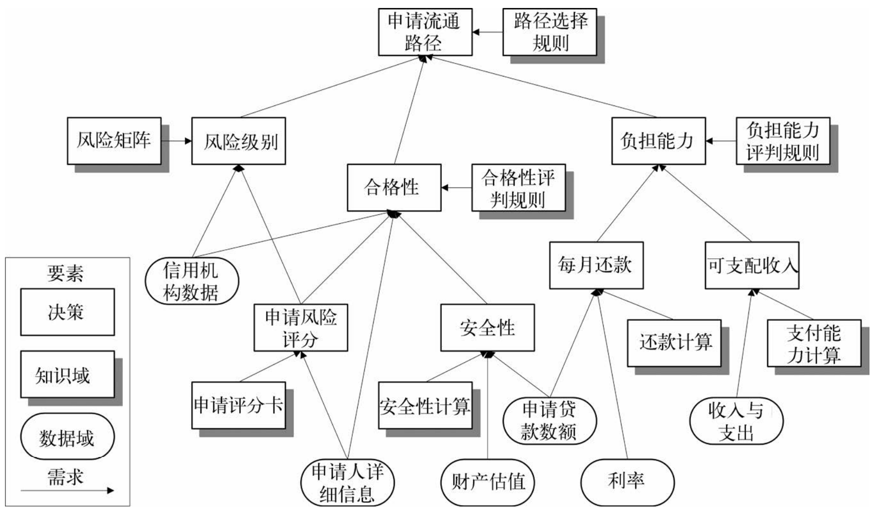


图 4-1 一个简单的 DRD

输入一组数据（数据域及其他决策的结果），每个决策会产生一组结果（决策值）。用于执行决策的业务逻辑都被封装在一个或多个知识节点中：决策节点使用这些知识作为输入来产生结果。

随着项目的推进，决策节点将被描述得愈加细致。最初的口语化描述包括：

- ❑ 名字：一个用于标记决策的短名词（例如“风险级别”）。
- ❑ 问题和答案：一个刻画决策的自然语言问题（例如，“这类申请通常有什么风险？”），以及为该问题定义的一组答案（例如，{RG1，RG2，RG3，RG4，RG5}）。
- ❑ 描述：决策制定逻辑或流程的一个简单定义（例如，“风险级别是使用风险矩阵对信用机构数据和申请风险评分进行综合评估之后得到的”）。

知识节点表示决策所需的业务知识域。知识可能以人类专家、印刷文献等形式存在，也可能不存在。第 3 章描述了此类知识在一个决策系统中的一些可执行方式，包括规则集、表、算法和分析模型等。

注意，我们可在尚未发现任何具体规则或者尚未定义任何详细计算时，就确定一个项目所需的知识域，也可以实现对项目大小及其复杂性的评估（例如，规则的数量、表的大小、评分卡及特征的数量等）。

数据节点表示决策所需的数据域，数据可能是：

- 与要处理的案例相关的特定数据（如申请、账户或索赔数据）；
- 与特定案例相关的上下文数据（如现行基准利率）。

再次声明，现阶段我们只需确定决策的数据域，并非所有具体数据项，尽管此时我们能够大概确定决策需要多少条数据项。

箭头代表需求：它表示制定一个决策需要使用某条信息的这样一个事实。需求并不是条件性的，若制定一个决策始终需要使用一条信息，此时连接一个箭头。

决策需求图表

DRD（决策需求图表）是一个网络图，它展示了在制定一个或多个决策时所需要的信息。

DRD 包含三种类型的节点：决策、知识和数据节点。

决策点代表了需要制定的决策。

知识节点代表决策需要的业务知识域。

数据节点代表决策需要的数据域。

箭头代表一项决策对于某条信息的依赖关系。

4.2.2　用途

DRD 的主要用途将在第 5 章中详细描述，此处仅进行简单介绍。

节点的定义对其自身直接有用。

- 决策节点为决策机制的功能性需求提供了一个清晰而简洁的定义。

- 知识节点确定了项目过程中所需获取或建模的全部知识域。
- 数据节点确定了决策服务实施及建模所需的全部数据域。

节点之间的箭头对于 DRD 的设计具有重要意义，它主要表示以下项。

- 数据流：若决策在系统组件内部被执行，而某些所需信息源却在系统组件的外部，此时需要构建一个数据接口。
- 任务执行顺序：若制定 B 决策需要使用 A 决策的结果，那么 A 必须在 B 之前执行。

选定一组节点并在其周围画一条界限，即可对界限内及界限外资源有一个非常清晰的定义，而跨过界限的箭头连接则暗示此处需要一个接口。这就使得以下工作可以实现。

- 确定项目范围：在所有将在项目内部实施的节点周围画一条界限。
- 分配系统功能：在众多决策服务和/或其他系统组件之间画一条界限。
- 迭代开发：将节点划分到不同的项目增量中。
- 项目资源分配：将节点分配给各个团队或个人。

图 4-2 所示为如何通过在 DRD 中画一条界限来确定项目范围。该示例展示了一个十分常见的场景：所有决策节点和知识节点都在界限之内（即决策服务必须执行已被定义的整个决策制定流程），而所有的数据节点都在界限之外（即决策服务所需数据必须全部由客户端基础设施提供）。

图 4-3 展示了如何使用 DRD 定义迭代设计和开发中的项目增量，例如使用统一软件开发过程（Rational Unified Process, RUP）管理项目。此处示例将决策节点和知识节点划分为三部分项目增量，每部分增量提供不同功能。决策需求分析（DRA）允许执行此类项目规划，从而使得三个增量之间是松耦合的，即增量之间通过少量几个箭头相连。

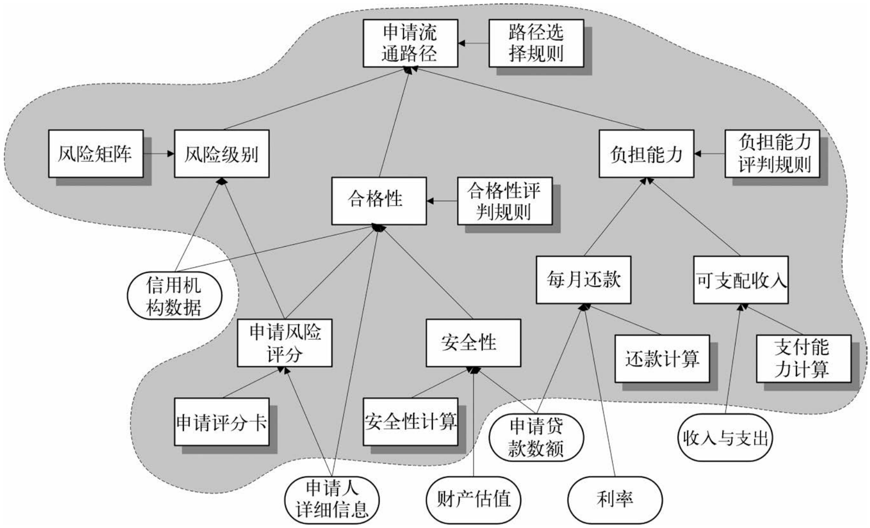


图 4-2　在 DRD 上确定项目范围

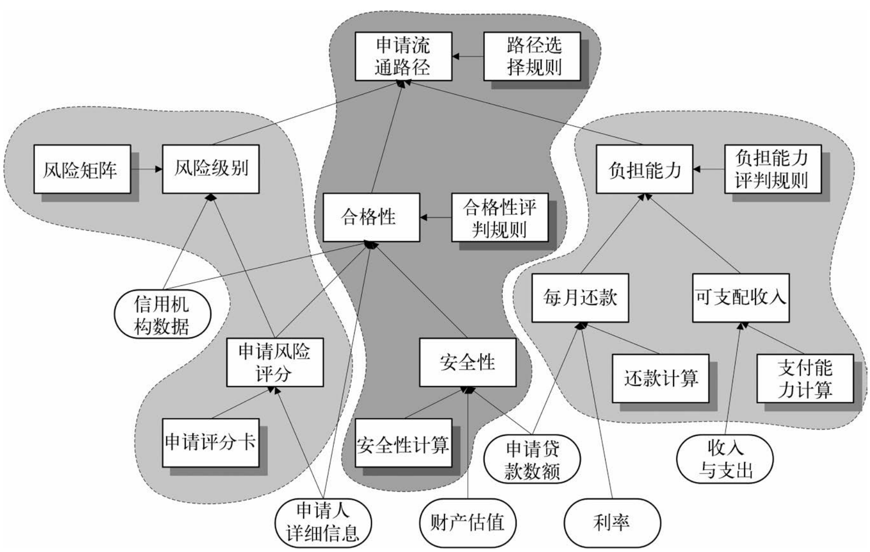


图 4-3　在 DRD 上划分项目增量

尽管十分简单，这些图表却非常有效。使用这些图表，参与项目的所有人员：项目赞助者、业务领域专家、分析师、设计人员、开发人员以及测试人员，均可通过简单的一瞥就能看清决策机制的范围和结构，并可围绕其展开讨论。这些图表的使用还允许人们无需参考任何技术或结构文档就可以指着图表说“决策制定

流程的这个部分……”。

DRD 是通过使用一种简单的、结构化的、被称为 DRAW 的研讨会技术生成的。

4.3 DRAW

DRAW 是一类结构化的研讨会技术，它可为一组决策服务定义决策需求并使用 DRD 对决策需求进行存档。一般情况下存在两种常见的 DRAW 应用场景。

(1) 它可作为一个完整的活动被执行，为可能的知识自动化项目确立所需资源（例如，确定可行性、优化备选项目或为多次项目交付生成一个线路图）。

(2) 它可在一个项目实施的初始阶段被执行，为决策制定流程定义需求（如瀑布模型中的需求分析阶段或 RUP 中的开始阶段）。

在 DRAW 上确定的需求会被记录在一个包含一个或多个 DRD 的文档中。

4.3.1 资源

无需任何特殊的工作环境或软件设备，DRAW 可在任何一间带有活动挂图、白色书写板及投影仪的安静的会议室举行研讨会。若参与者绕桌而坐，而非像听报告一样面对分析师而坐，则会显著地改善与会者之间的交互。若没有可用的编辑工具，可在研讨会期间先将 DRD 图表画在白板上，会后找一个常用的图表编辑工具规范的画下来。

DRAW 的成功实施完全依赖于参会者的才干与贡献，因此，有一个优秀的团队是非常重要的。以下人员应该全程参会。

- 决策分析师：促进研讨会的展开，分析讨论结果，将研讨会上达成共识的需求记录在包含一个或多个 DRD 的支持文档中。
- 抄写员：通过持续记录研讨会上达成的共识、决策所需信息及会上的出色表现等，为决策分析师的工作提供支持。

- 业务领域专家：贡献专业知识，代表业务流程用户，帮助提高业务流程的用户接受度及参与度。
- 业务分析师：负责业务流程重设计以及为业务流程自动化定义功能性需求。

该核心团队同时应得到以下人员的部分参与支持。

- 项目赞助者：负责定义最终项目目标及成功标准。
- 项目经理：负责项目规划、项目日常管理、联络及报告，以及提供项目所需资源。
- 决策服务架构师：负责设计决策服务实施已达成共识的决策需求。
- 技术架构师：为项目的实施提供技术上下文相关的内容（例如，整体系统架构，一般 IT 需求或标准，可用的数据资源和接口，将被替代的遗留系统）。

决策分析师将主持整个研讨会，引导决策需求分析并生成分析结果，大部分输入来自于领域专家及业务分析师。理想状态下，技术架构师应随时待命，同时必须参加任何与数据源及接口相关的讨论。项目赞助者和项目经理应在项目最初阶段以及在制定任何与项目相关的最终决策时出现。

4.3.2　日程安排

相比于大型研讨会，DRAW 更适用于小型团队（6 到 12 个知识型人才）。通常情况下，我们会以一个或多个全体研讨会作为 DRAW 的开端，首先制定出顶层需求，然后在必要的时候与其他各个子团队或个体专家进行更为细致的讨论分析。此时并不需要同每个人都探讨整个网络的结构：每个个体的专业领域知识或责任可能与网络中的不同区域相对应。但是，在多个人员间进行交叉检查却是一个好主意。首先，个体可能会遗漏某些数据项；其次，不同人员的意见分歧可能会揭露某些需要阐明的重要业务问题。在结果被分析和存档之后，需在最后一次总体会议上展示 DRD 的绘图，以达成一致意见。

尽管每天执行一个很高层次的 DRAW 是有可能的，但通常情况下，一个需要持续几天、多次研讨的 DRAW 则更为典型。对于一个包含单个业务过程的典型项

目，一个很好的方法就是预留一个星期，并为会议延期或后续工作预留 2~3 个半天的时间。这将为那些有其他时间安排的业务领域专家提供更多的方便。

要灵活。有一个预先安排好的日程是非常有用的，但你不可能准确预测决策制定的哪个环节将会需要更多讨论，因此，一般情况下你可能并不会完全遵循这个日程。随着网络的结构逐步被揭露，你可能会发现还需联系和拜访额外的人员。因此，事先考虑清楚参与人员的范围也是十分重要的。有一次，我到南非的约翰内斯堡，希望与六七个专家进行一个短小的 DRAW 会议，其中包括一个处理单一零售信贷产品的发放流程的延伸会议。到达之后，我发现 40 个人挤在一个很大的房间里，期待有一个面向全企业范围的、涵盖所有公司产品系列的业务重设计。这就有点太过灵活了。幸运的是，我喜欢和南非人共事。

4.3.3 方法

一般而言，DRAW 包括这样顺序化的 5 个阶段。

(1) 找到决策点。
(2) 定义顶层决策。
(3) 分解决策制定流程。
(4) 详细地描述所有节点。
(5) 明确决策服务的界限。

我们将在以下部分详细描述这些阶段。当然，很少会出现运行如此流畅的研讨会：有些情况下，需要你对前一阶段内容进行回溯并对其结果进行修改或增加细节。这些都不是问题，只要能遵循这些基本原则即可得到合理且完备的最终结果。

DRAW 的 5 个阶段
(1) 找到业务流程中的决策点。
(2) 定义顶层决策。

(3) 分解决策制定流程。

(4) 详细地描述所有决策、知识和数据点。

(5) 明确决策服务的界限。

阶段 1：找到决策点

我们将从业务流程以及组织实施业务决策自动化项目的目的开始讨论。试着从驱动者的角度进行思考：赞助者希望如何改变他们的业务？他们希望从决策管理中获得什么收益？

详细调查业务流程中所有需要决策服务自动进行决策制定的节点，它们即为决策点。为每个决策点指定一个名字并明确：

- 决策点执行任务时所处的业务流程阶段；
- 该阶段要制定的决策类型；
- 决策服务在决策制定流程中的预期作用；
- 任何用户或其他系统组件在决策制定流程中的作用。

使用业务流程流或工作流程图对这些结果进行存档。需要将业务流程描述到一定的细致程度，从而使得每个决策点都是流程中的一个原子任务或活动，仅包含负责自动执行决策的决策服务。流程与其他代理用户或系统的所有交互，必须在决策点之前或之后的单独任务中发生。这使得我们可对数据流需求进行分析：决策点使用的所有数据必须在该点之前就收集完毕，决策点输出的结果只能在该点之后的流程中被使用。

图 4-4 展示了阶段 1 结束时可能出现在白板上的一个图。该流程流遵循 BPMN 2.0 的一般规则，包括两个以业务规则任务形式表现的决策点：调用信用机构之前及调用信用机构之后，分别对应发放流程模板中的信用机构策略和申请流通路径（见图 2-8）。大多数真实的业务流程都比图 4-4 所示的流程更为复杂。但也有些比它简单，例如，一个每月触发一次且仅仅用来收集数据、调用一个决策服务并对数据结果进行存储的批处理。

第 2 章围绕决策自动化讨论了一些问题，这些问题表明，在引入自动化决策机制时可能会涉及其周围业务流程的重设计。因此，该阶段所需的讨论时间可能十分多变。我参与过一些业务流程固定、决策点预定义好并可在几小时内完成确定决策点任务的项目，也参与过一些将确定决策点任务作为整个流程重设计的一部分的项目，这往往需要花费几个星期的时间。

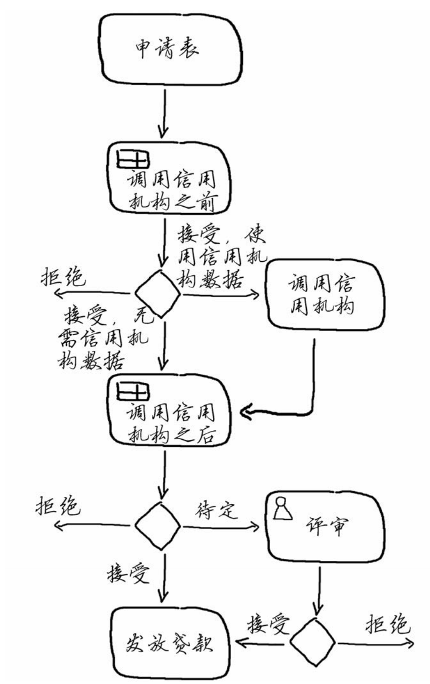


图 4-4　DRAW 阶段 1——决策点

阶段 2：定义顶层决策

为每个决策点确定其决策服务所需给出的主要决策，并使用最普通的措辞进行描述。此处主要决策是指会在业务流程中实际用到的决策（例如，路径选择或案例处理）。若流程中有多个决策点执行类似决策，试着根据上下文对其进行通用化处理。有时会发生这样的情况，某些不在该阶段的决策服务试图针对特定服务分配决策，此时仅保留那些必要的决策点。

为每个决策定义一个名字，并同时明确下述特征。

- 一个刻画决策的问题，该问题以自然语言描述且带有一组定义好的答案。
- 需要与问题答案一同返回的任何其他结果。
- 制定该决策的所有决策点，以及不同决策点上制定的决策有何变化。

现在白板上可能会变得有些混乱，它看起来应该如图 4-5 所示。在该示例中，有 4 个主要决策（每个决策点上有 2 个决策），这些决策结果将被用于实际的决策制定流程中。

- 合格性及所需信用机构决策，调用信用机构之前决策点给出的决策，决定了申请的即时流通路径（直接通过、拒绝或调用信用机构）；若调用信用机构，此时需确定，可通过调用信用机构获得哪些数据。
- 流通路径决策确定了一项申请的后续去向（接受、待定或拒绝）；若申请被接受，发放贷款决策需决定向申请人发放的贷款数额。

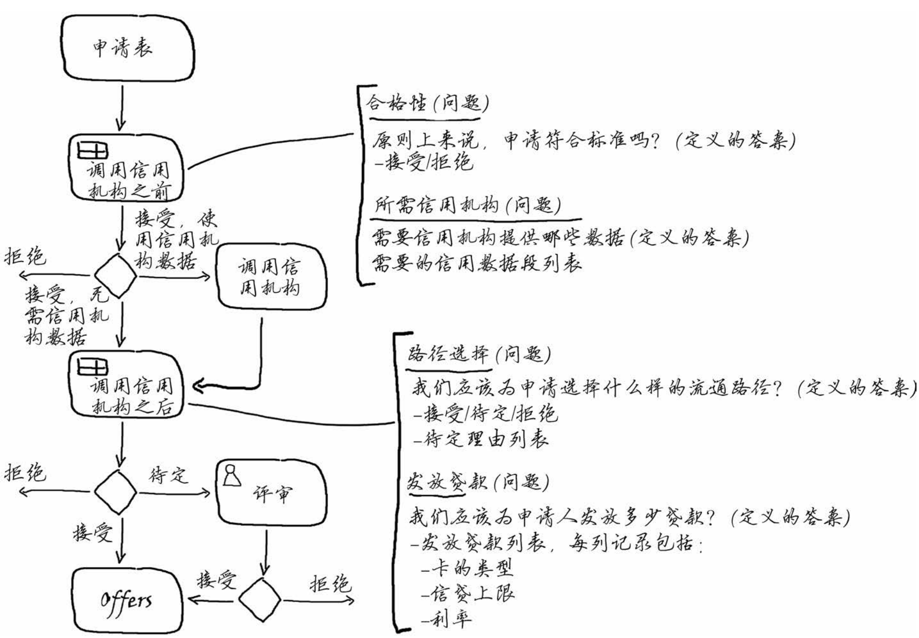


图 4-5　DRAW 阶段 2——主要决策

阶段 3：分解决策制定流程

现在可以构建我们的 DRD 了。首先，把在阶段 2 中确定的顶层决策表示成一个个不相连的方框，画在白板上。在我们的例子中，这些顶层决策分别是合格性、所需信用机构、流通路径和贷款发放。使用这样一个迭代过程——选择一个决策并提问“若要制定该决策，都需要哪些信息”，然后充分探索下述可能性：

- 业务知识域（已存在或尚未被创建）
- 数据域（描述案例或其上下文环境的特定数据）
- 其他决策的结果（子决策）

根据决策需求，逐渐地向 DRD 中添加知识、数据或决策节点，然后将其连接到你所强调的决策上。以定义顶层决策的方式对添加到 DRD 中的新决策节点进行定义。随后它们可按顺序得以强调。注意，由于这是一个迭代过程，每次仅分析一个决策，分解一个大型决策制定流程及其需求并不会变得更加困难，只是花费更长的时间而已。

例如，若提问“你需要什么信息来判断一个申请是基本合格的”，领域专家也许会回答：“嗯，我们有许多基于申请人年龄、是否失业等申请信息的业务策略。哦，对了，其中有条策略是关于最低风险评分的，因此我们也需要知道这条信息。”这就需要在 DRD 中增加一个策略规则知识节点，增加一个申请信息数据节点，同时增加一个风险级别子决策的决策节点，然后将它们连接到合格性决策节点上。

然后可针对申请风险评分子决策询问同样的问题：“你需要什么信息来判定申请人的风险评分？”专家会告诉你，我们可使用一个仅依赖于申请信息的评分卡来判断一个申请人的风险评分。所以，只需增加一个评分卡知识节点，并把已创建的申请信息数据节点连接到风险评分决策节点上。此时，我们的 DRD 应该如图 4-6 所示。

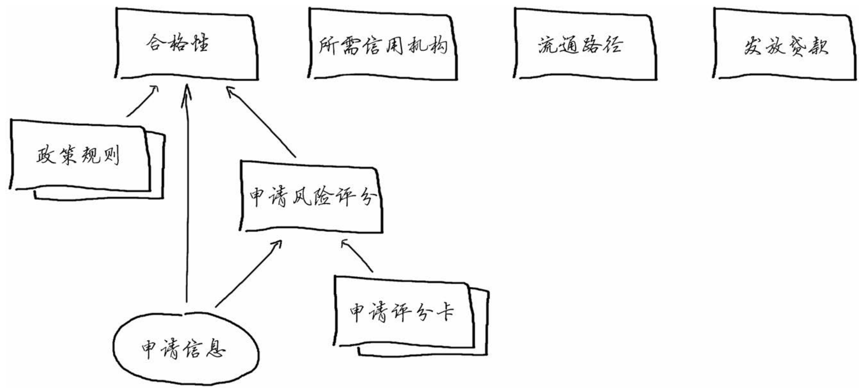


图 4-6　DRAW 阶段 3——DRD

现已无需再为合格性决策提供任何其他决策信息，我们可以把注意力移到下一个主要决策：所需信用机构。你被告知："我们购买了所有申请人的信用机构数据，那些被拒绝购买的选项除外。我们为每个申请人购买了一组标准信用数据，但若申请人的申请风险评分处于合格边缘，我们也会购买其信用评分。我们有张表可以告诉你，不同类型申请人的哪些风险评分的分数段是处于合格边缘的。"这告诉我们，所需信用机构决策依赖于合格性和风险评分决策，它需使用申请信息，并需使用 CRA 表中涵盖的业务知识。现在，这个 DRD 看起来如图 4-7 所示。

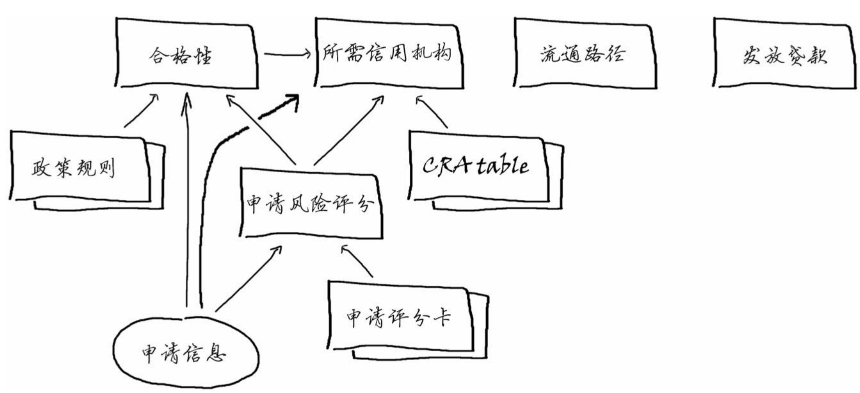


图 4-7　DRAW 阶段 3——接上面的 DRD

慢慢地，你的 DRD 会扩展到整个白板，这看起来可能有点乱（使用不同颜色进行标记可能会有所帮助）。但当整个流程结束之后，你会得到一个完整的 DRD，

包含所有需被确定的决策、决策包含的知识域及数据域。最初确定的、相互独立的四个决策有可能已被直接地（如合格性和所需信用机构节点）或通过共享的子决策间接地连到了一起。

白板的大小可能会成为 DRD 规模的一个限制因素，所以有些情况下，为每个决策点画一个单独的 DRD 图表会更为方便。只要你知道，原则上它们只是一个大的 DRD 即可。同时，一个决策点上制定的决策可作为后续决策点的决策输入。重要的是，此时第二个决策点应该将决策输入记作决策之间的依赖关系，而非数据源。设想，在我们的例子中，若为两个决策点分别提供一张图表，那么流通路径决策就需基于申请的风险评分，并参考高风险申请案例的规则。此时，调用信用机构之前的风险级别决策节点，也应该出现在调用信用机构之后的 DRD 中，并连接到流通路径决策节点上。此时，我们应该将该节点（风险级别决策点）作为一个连接器，而非复制它所有的依赖关系（我使用一个*号来标记）。

当执行该分析时，决策分析者必须不断地判断决策制定流程分解过程中活动粒度的大小。一般来说，当你能够明确每个知识域是如何作为一类单独的业务知识被执行时（如第 3 章中所讨论的），你就得到了一个完整的分析。这也是我在第二章中的结束语处所说“带有预见性的自顶向下”的一种方法的意思。决策分析师本身并不需要是一个设计者，但是需要对设计者所使用的方法有一个基本的了解。当然，这些方法依赖被使用的 BRMS 的具体特征。若在执行完 DRAW 后仍未选择具体的 BRMS，那么第 3 章中描述的框架则是一组很好的假设。

阶段 4：详细地描述所有节点

现在，应该重新访问 DRD 中的所有节点并对其进行逐一详细探讨。

针对每个决策节点，全面地调查如何使用相关知识和数据实现其决策。检查目前描述过的所有标准场景的例外情况。若有需要，在一个涵盖所有场景的、更为细致的描述上达成一致。

检查每个知识节点，并记录：

- ❑ 知识来源（例如，文档、电子数据表、个体人员、遗留系统或分析方法）

- 节点规模及复杂性的评估结果（如近似的规则数量和计算页数等）
- 谁来负责提供该知识（最好有一个具体的名称）
- 知识节点的维护需求，包括谁来更新维护知识（业务人员还是 IT 人员）以及维护频率
- 任何其他重要的背景信息（例如，作为知识源的指定个体的可用性的限制，以及是否存在在规定的项目时间内引入新型决策流程的计划等）

检查每个数据节点，并记录：

- 数据来源（纸质文档、输入界面、Web 服务或数据库）
- 节点规模及其复杂性的评估结果（例如，近似的数据项数量）
- 谁来负责定义数据接口
- 任何其他重要的背景信息（例如，数据源的准确性和完备性，在规定的项目时间内可能出现的任何改动）

这个过程包含一些重复工作，深入探讨了阶段 3 所收集的需求。这主要是考虑到，它是一个检查你所记录的信息是否完整可靠的好机会。

注意，该阶段的目标并非收集实际知识（如具体的业务规则或算法等），而仅仅是为一个决策描述其所需的知识域并为后续发现确定数据来源。然而，专家们有时可能会使用某些具体的示例规则作为解释内容的一部分，如果是这样，应该把这些示例规则记录下来。类似地，当对数据节点进行检查时，目的主要是确定所需数据域，而非确定一个完整的对象模型。

阶段 5：明确决策服务的界限

已经确定了决策制定流程的结构并对决策所需的支持信息进行了存档之后，第 5 阶段我们要确定哪些节点应作为决策服务的一部分，而哪些节点将被留在决策服务之外。不要试图在该阶段决定应该提供多少项决策服务或如何在决策服务之间分配决策，这是一个设计问题，在此我们要做的仅仅是收集需求。只需确定哪些节点将被服务执行。这将确定决策服务的一个“范围界限”，如图 4-2 所示。

用于确定决策服务范围界限的一般规则如下。

- 顶层决策节点将在决策服务中被实施（这是我们的初始假设）。
- 大部分子决策及它们的知识支持节点将在决策服务中被实施，但有些可能需在其他决策制定流程中实施（可能是暂时的），如在遗留系统中或由某些高层人员实施。在调用决策服务之前，应收集其所需的所有外部决策的结果，并将这些结果以数据的形式传递给决策服务。
- 大多数的数据节点都作为决策信息源，被置于决策服务的外部。但方便起见，若无需被其他系统访问，这些节点将被置于服务内部实施。

界限分析可能会涉及某些临时的架构层面上的决策，因此，决策服务界限可能会在随后的设计过程中发生改变。由于这个问题，最后一次的 DRAW 会议将展示全部分析结果，这需要所有利益相关者的参与，同时也要求所有人在 DRD 及其范围界限的问题上达成一致。

一旦在最后的 DRAW 上针对所有问题达成共识，即可将 DRAW 的结果记录在一个需求文档中。包括：

- 业务流程流及其决策点
- 主要决策
- DRD 及其所有节点的支持描述
- 范围界限

这是一个轻量级的文档，大概在 20 页左右，但一旦达成一致，就是一个非常精确的表述决策服务自动化范围的文档。该范围文档之所以如此重要，是因为它基于对决策制定流程结构的细致分析，而非仅对决策数量的模糊估计。对决策结构的清晰理解将有助于实现精确的项目评估、稳健的项目规划、简洁的设计、结构化的知识发现，使用可重用代码高效实施项目，避免范围扩散并减少蔓延风险。下一章将详细描述如何实现这些目标。

给决策分析师的 10 个小贴士

(1) **制定一个计划。**很少有比邀请了一屋子的主题事务专家（Subject matter experts, SME），却不知道要做什么更为尴尬的事情了。DRAW 是一个结构化的流程，它可以提供很多帮助。同时，也可通过业务线或流程来打破业务领域之间的壁垒。在研讨会之前，最好创建一个日程表并让 SME 传阅，使他们了解每个会议部分将要讨论什么内容，从而能够做好充分准备并携带每次所需的相关材料。

(2) **解释业务流程。**为了保证效果，SME 们需明白研讨会上发生了什么事情，在朝着什么方向发展，目标是什么。向 SME 们解释 DRAW 的原则与方法，并描述预期结果。让每个人都理解研讨会预期结果的形式以及结果所需的详细程度是十分重要的，即，我们希望 SME 们可以清晰地定义所需制定的决策，而非提供任何具体的业务规则。

(3) **采取控制，保持控制。**坚持遵循 DRAW 的整个流程。在一定程度上，保持灵活性是十分重要的。整个过程中不可避免地会出现走弯路的情况，但是一定要使 DRAW 在掌控范围内。SME 们往往是具有很强个性的人，所以某些情况下要努力别让他们把会议带偏离主题。决策分析师的角色就是要促进并控制讨论，而非简单地记录会议进展，同时，保证每次会议都能达成其目标是决策分析师的首要责任。

(4) **协作。**不要简单地对会议上描述的已有的业务流程以及业务逻辑进行存档。决策分析师的任务是帮助组织重构其业务流程及决策机制，从而使组织从决策管理中获益，所以你需要质疑、挑战并提出建议。若你在其他项目或领域中累积了经验，提出“有没有想过可以这样做？”的问题也是十分有价值的。

(5) **坚持弄清楚。**如果你认为某件事情完全不合理，那么有可能是你并未完全理解它，或是某些环节出错了，也可能是因为出现了前后不一致的情况。若发生上述任何一种情况，都需要弄清楚问题。不要害怕提出一些愚蠢的问题：你是分析师，并不是 SME，你的“无知”有助于更好地解决问题。

(6) **化解矛盾**。如果你确实发现了一些不一致性或意见分歧，把它们当作问题记录下来，如有必要，也可将其返回给组织请求上级帮助解决这些争议。一项决策服务仅能执行一条连贯的公司策略，所有利益相关者都必须在这点上达成共识。

(7) **反复检查所有规则**。从不同角度考虑每个问题，检查所有的一般性规则，并思考是否存在例外情况："这总是正确的吗？从未发生过例外吗？"在继续下一个话题之前，不断地提问，把你对需求的理解重复给 SME 们并寻求他们的确认，从而确保你的理解正确。

(8) **记录所有相关事宜并存档**。可能存在许多这样的情景，你认为"这太明显了，我不需要记录它"，然后就会发现，一旦离开会议室，你就会忘记（特别是当你需要同时处理多个项目时）。用你正在使用的任何工具进行记录，如果你正在使用一个白板，就使用标签或日期对记录进行标记，并把它拍下来。

(9) **照顾好自己**。参加一系列 DRAW 会议可能需要你连续几天全神贯注地参与，这会使人筋疲力尽。如果累了或者心里不舒服，你将不能有效地完成自己的工作，这是从我自身的经验中得出的结论。

(10) **做自己，享受工作**。每个人都有自己的工作方式，咨询顾问是人类交互的一种方式，它需要坦诚地、放松地、自然地享受这个过程并与 SME 保持良好互动。良好的工作方式有两个优点：首先，它会使得专家们保持放松，鼓励他们更为开放地讨论问题，并消除紧张的气氛；其次，你会获得更多的乐趣，工作应该是快乐的。

4.4 参考文献

1. Object Management Group, *OMG Unified Modeling Language* (OMG UML), Superstructure Version 2.3 (2010) OMG Document Number: formal/2010-05-05, www.omg.org/spec/UML/2 .3/Superstructure.
2. B. Von Halle and L. Goldberg, *The Decision Model: A Business Framework Linking Business and Technology* (Boca Raton, FL: Auerbach Publication, 2010).

第 5 章

DRA在知识自动化项目中的应用

截至目前，我们已经重新建模了业务流程，并确定了一组将被自动化（通过在决策服务中封装组织业务知识）的决策点。同时，我们使用决策需求分析（DRA）定义了将在这些决策点上实施的决策制定流程的结构。因此，我们已经基本掌握了知识自动化的所有高层次功能性需求，具体表现为业务流程流以及包含支持文档的决策需求图表（DRD）。现在，我们将设计、实施并部署一组决策服务来满足这些需求。

此类项目可能充满了各种各样的问题，而大多数问题的产生是由于缺乏明晰的决策结构所致。通过使用 DRD 中展示的决策流程结构作为模板，本章将提供一个项目结构、一个知识发现结构、一个设计结构以及一个实施结构，从而提供一些能够缓解甚至完全避免这些问题的建议。

5.1 DRA在项目管理中的应用

DRA 揭示的决策结构对于保障知识自动化项目的良好管理具有以下几个重要作用。

- 为项目范围有一个明确的合同性定义提供了基础。
- 为工作指标和时间指标的估算提供了一种度量方法。
- 表明了项目规划和资源规划的结构。

- 允许定义迭代式项目管理或敏捷项目管理中的增量。
- 允许对从项目交付成果到初始需求的每个阶段的全程追溯。

由于上述优势，DRA 于 2005 年被 FICO 采用并被纳入其基于统一软件开发过程（Rational Unified Process）[1]的方法论 FIRUP（Fair Isaac RUP）中，作为决策需求定义和项目管理的标准工具。自此之后，它已被应用于世界各地许多不同的知识自动化项目中，并取得了巨大成功。

5.1.1 自动化范围界定

本质上，决策需求分析研讨会（DRAW）是一个确定项目范围的活动，因此应在知识自动化项目中尽早落实。（如第 2 章所述，决策服务的需求应当作为业务流程重设计的一部分一并考虑。）

DRAW 的讨论结果应被记录在一个文档中，并用于定义将被自动化的决策范围。这种自动化范围界定文档的确切作用取决于研讨会的上下文环境、采用的方法以及参与各方，例如：

- 可在业务案例开发初期为项目融资提供一个可行性及成本的评估；
- 在客户和供应商之间形成一个合同性的文件，指定后续开发项目的供应范围；
- 可以是项目开发的第一个阶段（瀑布式开发的需求阶段，RUP 的初始阶段）产生的一个高级需求文档，用于明确项目范围和促进项目规划。

DRAW 的实施时间越早，组织从范围界定文档中可获得的利益就越大。正如需求文档一样，DRAW 的结果将十分有助于项目规划和管理，但仅限于一个范围已被广泛达成共识的项目。若在项目开始之前，将 DRAW 作为一个单独的范围界定工作展开研究，即可用其结果计算项目成本并支持项目谈判。若用于可行性评估，则 DRAW 可估算出一些争取项目资金的机会，从而为项目可行/不可行的决策提供支持。

我在上述所有场景中均应用过 DRAW。下面是我在 FICO 公司曾参与的一些

示例项目。

- **可行性评估**。作为给英国健康保险公司提供咨询服务团队的一员，我审阅了他们大量的业务流程，从中识别实施自动化决策机制的潜在机会，并使用决策收益定价法对决策自动化将会带来的效益进行了评估。然后，我组织大家进行 DRAW 并围绕最具潜力的方案展开讨论。讨论结果最终被用于判定可行性、估计开发成本以及制作项目实施路线。
- **合同范围界定**。最近参与的一个项目被分成了两个独立的合同，该项目的内容是帮助一个英国建筑协会实现组织的自动化管理。第一份合同使用 DRAW 定义了项目自动化的范围，并设计了一个技术架构。这些结果随后被用于指定项目第二阶段任务的合同声明，并提供了一个更为准确的项目实施成本评估。
- **定义高层次需求**。一个南非家具零售商聘请 FICO 为其信贷发放过程设计一个大型的企业级流程自动化项目。该项目首先使用 DRAW 为每一条业务流程流确定其所需的自动化范围，然后将 DRAW 的讨论结果记录在一个功能性需求文档中，该文档随后被应用于决策服务开发中。同时，决策需求分析允许开发一个跨越所有流程线并可在大量决策结构中被重复使用的通用对象模型。

无论 DRAW 处于怎样的环境或场景，其生成的结果文件在表现形式上本质都是相同的。同时，无论该文档的作用是什么，它应该至少包含以下部分。

(1) 业务上下文：项目背景，包括决策自动化目标以及项目成功的判定标准。

(2) 决策点：业务流程中的一个描述，受业务流程流图支持，流程图指定了那些需要自动化服务执行决策的点，并确定了该点所要执行的主要决策。

(3) 决策结构：流程中每个决策点的 DRD，受所有节点的文字定义支持，包括对于知识节点、数据节点的大小和复杂性的评估。

(4) 自动化范围：对于可被自动化的决策的范围声明，由 DRD 上的一组节点定义，并表现为一个带边界的 DRD 子图，如图 4-2 所示。

自动化范围界定文档

自动化范围界定文档：

- ❑ 定义了一个决策自动化项目的需求。
- ❑ 可用于可行性评估、合同范围界定，或作为一个项目的需求文档。
- ❑ 始终包含以下内容。

(1) 业务上下文描述：项目的背景、目标及成功标准。

(2) 决策点：业务流程中的某些点，这些点上将调用决策服务并将制定流程中的主要决策。

(3) 决策结构：流程所需决策点的 DRD，受所有节点的文字定义支持。

(4) 自动化范围：将被自动化的决策范围的声明，通过在 DRD 上绘制节点边界来定义。

这是一个十分简明扼要的文档，但却提供了决策自动化需求的明确定义，并允许对自动化范围进行严格控制。这即表示，该文档必须在开展任何后续工作之前接受项目赞助者审查并签署。这并不是为了防止它在项目中途被改变，而是为了任何改变均可被正确管理。基于 DRAW 结果中定义的项目范围，任何修改范围界限的建议都可根据其对项目的影响进行评估。

例如，每一条单独的业务实施规则都应用于边界范围内的一个决策，它属于边界范围内的一个知识域，并仅使用边界范围内数据域所提供的数据。若发现某一规则不符合这些限制，应随后执行已达成共识的变更控制流程，考虑扩大范围成本（使新范围能够包含新的决策、知识域或数据域）。这看起来似乎有些苛刻，但它却是控制项目范围蔓延的一种十分有效的手段。在业务规则项目中，范围蔓延是一种十分常见的问题。若已提供一个全面的决策需求分析，则极少需要这样的范围变更。

5.1.2 项目评估

一旦针对项目范围达成共识，业务流程、DRD 以及范围边界即可提供许多度

量指标。这些指标可对实施决策服务的一个项目所需的工作量进行评估，不同指标具有不同的含义。

- 决策点的数目表示将被部署和测试的决策服务实例的数目。
- 边界范围内决策节点的数目表示将被实施和测试的单独决策的数目，它同时表示决策服务建模及其返回的输出数据量。
- 所需数据节点（与边界范围内决策节点相连）的数目和复杂性表示将被建模并提供给决策服务的输入数据量。
- 边界范围内知识节点的数目、类型、复杂性表示将被发现、实施和测试的业务知识量。
- 范围内连接决策节点的箭头数量表示了决策流程的整体复杂度（连通性）。

这些指标可用于创建预测项目工作量的模型。总体来说，通过使用不同指标的组合，可实现各种复杂程度的模型建模。对于项目可行性评估，即可使用一个简单模型对整个项目成本进行粗略估计；这单纯地基于决策点和知识节点（也是目前为止最为重要的指标）即可实现。对于详细的项目规划，可使用更为复杂的模型对特定项目任务所需的工作量进行预测（这将在下一节进行讨论）。

不要期望能够直接利用现成的模型。在一个给定的范围中进行自动化，所需的工作可能会根据组织的一些具体因素而变化，这些因素如下。

- 使用的项目方法，如文档标准和测试方法等。不同方法会产生不同级别的开销及不同级别的风险。
- 解决方案的架构，尤其是选择的业务规则管理系统（BRMS），部署的服务类型，以及提供的知识维护设施。
- 业务知识的领域。
- 主题事务专家（SME）及项目实施团队的研发经验。

因此，此处无法提供一组适合所有项目的标准模型系数。

但是，重点在于 DRA 提供了一种可被植入模型之中的定量评估项目工作量的

方法。当模型第一次被使用时，它将单纯地基于专业判断，因此评估结果可能会存在较高的不确定性，但仍可能比任何替代方法都要可靠。这种不确定性必然在项目异常中有所体现。由于组织通过执行许多知识自动化项目来获取经验，它可利用从已投入运行的项目中收集到的统计信息，将模型调整为适用于特定项目方法、架构、业务领域和团队的形式。

如果使用迭代或敏捷开发的方法，我们也可通过监测项目进展以及对比预期工作量与预期燃率，在项目中途对模型进行微调。

使用 DRA 进行项目评估

以下这些指标都可用于预测项目工作量：

- **决策点**的数目
- 范围内**决策节点**的数目
- 范围内所需**数据节点**的数目及其复杂性
- 范围内**知识节点**的数目、类型及其复杂性
- 范围内连接决策节点的**箭头**的数目
- 其中最重要的指标是决策点和知识节点。

模型可以使用已投入运行的项目统计数据或（迭代项目管理中）在项目中途得到的统计数据进行调整，并使模型自身可为特定团队所用。

5.1.3 项目规划

一个知识自动化项目可被视为一个在决策服务中封装业务知识的项目。这些项目的独特性在于其知识发现任务：所有需被自动化的知识都必须被收集、达成共识并编码成可执行的形式。该任务作为项目的主要组成部分，其所需工作量与设计和开发任务相当，甚至有过之而无不及。

知识发现是项目风险的主要来源之一，因为该过程是开放式的，并且若将其作为一个单独任务进行管理又过于庞大。这种风险与依赖知识发现的设计和实施任务有关。其风险解决方案为：*在知识发现任务开始之前*即使用 DRA 确定知识自

动化范围。通过在 DRD 上的每个知识节点完成特定的发现、开发和测试任务，在项目开始阶段即可产生完备且详细的项目规划。如先前所述，这些任务需要的工作量可使用 DRAW 产生的结果进行评估。

软件开发项目有两种基本模型：瀑布模型和迭代模型。DRA 对于这两种模型均可有所助益。

1. 瀑布模型

在瀑布式方法中，项目的实施需要经历如下一系列阶段，并顺序展开：

(1) 需求
(2) 设计
(3) 开发
(4) 测试
(5) 部署

瀑布式方法的核心思想为：在任何设计任务开始之前，收集、讨论并确定所有需求；然后在任何开发任务开始之前，完成所有设计。这一理念是基于这样的观察得出的：在根据需求进行设计之前，修复需求阶段产生的错误要简单得多，同理，在将设计付诸开发之前，修复设计阶段产生的错误要简单得多。因此，瀑布方法强调，在移到下一阶段前保证对于上一阶段结果的完整存档（又名“预先做大量设计”）。

瀑布模型在实践时也发现了许多问题，不过，主要原因在于人们很难在毫不改变需求的情况下完成项目设计和项目开发。设计和开发可能需要数月之久，因此需求很有可能会在这段时间内发生改变。同时，人们可以从设计和实施中吸取经验教训，特别是当所用技术是一项新技术时，他们从不同角度看待原始需求，并寻找不同方法创造机会以满足需求。所以在过去 20 年间，软件行业已经逐渐远离纯粹的瀑布模型，并开始尝试一些改进的方法，这其中就包括迭代模型。

一个纯粹的瀑布式方法根据大量知识发现的结果，对项目提出特定问题。本

质而言，编码过的业务知识是一组需求：它定义了系统应该如何运行，或者应该制定什么决策。但是，若所有设计和开发都需等待知识发现任务的完成，那么一个项目规划所需的时间将远比必需的时间要多，且并不能合理利用资源，因为知识发现团队（业务专家和决策分析师）通常是完全独立于项目实施团队（设计师、开发人员和测试人员）的。这两个团队完全可以并行工作。

可能现在你已经意识到，我并不喜欢在知识自动化项目中使用瀑布模型。但是，如果使用瀑布模型，DRA 同样会是一个十分有价值的工具，原因如下。

- 它恰恰在需求阶段确定了知识发现的领域范围。这将允许我们对需求阶段进行有限规划，而非开放式的知识发现。这同时确保了在该阶段结束前即可获得完整的项目需求。
- 它为项目设计提供了一个架构,允许项目基于 DRD 将其设计分解成多个功能组件。这同时确保了该阶段结束前即可获得完整的项目设计。
- 它明确了所有数据需求，降低了在开发或测试阶段需确定额外的整合决策点的风险。这可避免在需求及设计阶段进行大量返工。
- 它对于自动化范围界定文档中包含的将被自动化的决策范围提供了一个非常明晰的定义，可以以此为基础调整项目测试。

因此，DRA 缓和了瀑布模型最糟糕的特性，并大幅降低了瀑布式方法中经常出现的返工和溢出风险。

2. 迭代模型

迭代模型是对解决瀑布模型中遇到的问题的一种尝试，它将一个项目分为一系列小型项目或迭代增量。每次迭代强调一个软件生命周期的所有阶段——需求、设计、构建以及测试——来实现功能的增加。目前存在众多迭代开发方法，从逐步迭代到敏捷开发等，它们的主要区别在于迭代的长度以及增量的性质。近年来，迭代理念愈加普遍，业界同时也已开始尝试更短的迭代。

瀑布模型遇到的问题变成了迭代式方法的优点：每次迭代结束都为重新评估

需求和设计提供了机会，且通过连续迭代使得团队对使用的技术日臻熟悉。然而，瀑布模型的优点变成了迭代式方法的恶习：在项目开始时缺乏一套完整的需求，就意味着项目没有明确的完成标准，这可能导致项目范围膨胀、溢出，甚至出现最坏的情况，项目遭到废弃。

DRA 可解决上述问题。通过明确的定义项目增量以及完成标准，DRA 为迭代项目的规划提供了结构基础。基本原理为，项目规划的结构应该反映将被自动化的决策机制的结构（如 DRD 中所揭示的）。功能增量的定义十分简单，通过把 DRD 中的节点划分到不同的组中即可实现。这样划分的目的如下。

- 每个增量均包括一个完整的周期或一次迭代：每个增量的知识发现、设计和实施阶段执行的工作（如之前讨论所估计的）大致相同，并且在给定的迭代周期内是可实现的。
- 每个增量都会为组织提供一些不同的功能性效益，即每个划分中制定的决策在一定程度上仅在划分内部有用，即使最终的顶层决策处于另一个划分之中。
- 增量是松耦合的，即只有少数箭头连接的是处于不同划分中的节点。这通常意味着，每个决策节点应当与其所需的知识节点处于同一项目增量中。
- 增量与决策点之间存在一种简单、方便的关联方式，例如，一个增量涵盖了某些决策点上的所有决策，或（更常见的是）一个决策点被若干增量所实施。

图 5-1 展示了一个单一决策点（申请流通路径）所需的所有决策是如何被划分到一系列增量中的。我们的 DRD 示例在图 4-1 中被划分为 3 个增量。

增量 1：风险

- 知识发现：申请风险评分卡和风险矩阵
- 决策实施：申请风险评分和风险级别
- 数据集成：申请信息和信用机构数据

增量 2：负担能力

- 知识发现：还款计算，负担能力计算，以及负担能力评判规则
- 决策实施：每月还款，可支配收入，以及负担能力
- 数据集成：申请贷款数额，利率，以及收入和支出

增量 3：流通路径

- 知识发现：安全性计算，合格性评判规则，以及路径选择规则
- 决策实施：安全性，合格性，以及申请流通路径
- 数据集成：申请人资料，信用机构数据，财产估值，以及申请贷款数额

需要注意的是，划分揭示了：如果一个增量需要任何外部数据源对其所需交付的决策模块进行测试和集成，可将这些数据源都表示为由指向增量内部决策节点的箭头连接的数据节点。划分还表明了增量之间的依赖关系（例如，增量 3 中的申请流通路径决策取决于增量 2 中的负担能力决策）。这些数据和决策之间的依赖关系表明了项目规划中需考虑的项目增量之间的依赖关系。松耦合划分的目标之一即是要尽量减少这种依赖关系。

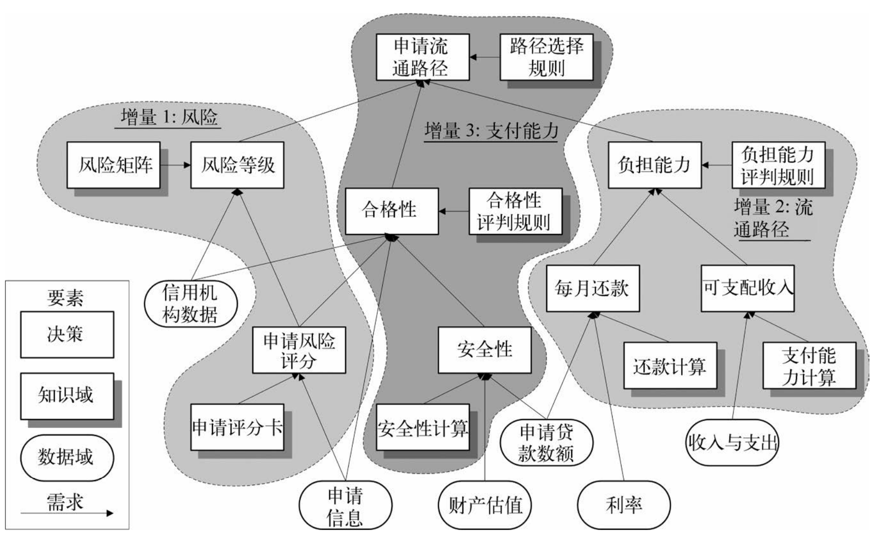


图 5-1　根据 DRD 划分项目增量

选择的方法越敏捷，增量就越小。我认为，大多数情况下，可加快开发速度的最小有益增量，至少需包括一个单一决策节点及其所有知识域。若其知识域十分庞大或复杂（例如，当规则集可能包含数百条规则时），还需将它们划分为多个知识域并置于不同的迭代中。当使用基于单一决策的增量时，我建议从 DRD 底部开始向上工作，这样会使得每个加速中决策所需的子决策已在前一个加速中实现。

然而，我偏向于更大规模的、可持续大约三至六周的迭代项目。这会为管理性质的项目任务，例如知识审查和注销，争取到更多的时间，同时允许知识发现团队和项目实施团队自然地相互磨合。请记住，业务专家还必须有一定空闲时间来完成他们的本职工作，专家全职借调到知识自动化项目是非常少见的。

3. 统一软件开发过程

一个重要的迭代方法是统一软件开发过程（RUP）[①]，如图 5-2 所示。

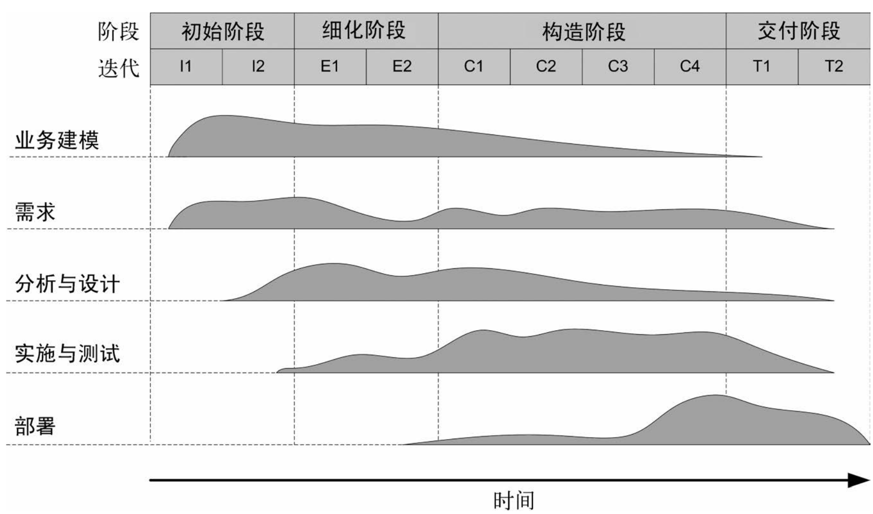


图 5-2　统一软件开发过程

在 RUP 中，每个项目被分为四个阶段。每个阶段包含一次或多次迭代，阶段末尾都有一个明确定义的里程碑来给出关键的项目决策。

① 摘自 Rational Unified Process: Best Practices for Software Development Teams，*Rational Software White Paper TP026B*， Rev. 11/01。

(1) *初始*。初始阶段的主要工作包括定义项目范围并确立其相关业务案例。范围定义通常涉及用例分析：确定该系统和其他实体（用户或其他系统）之间的所有互动，并使用用例描述这些互动。业务案例包含一个对于后续阶段的高级规划，该规划由以下几点要素组成：所有主要里程碑的实现日期，对所需时间和资源的评估，以及项目成功的标准。若业务案例结果并不令人满意，即可在此时终止项目。

里程碑：生命周期目标

(2) *细化*。细化阶段的主要目的是识别并消除最为显著的项目风险，尤其是技术性风险。这同时涉及分析问题域，构建合理的架构基础，并制定一个全面且细致的项目规划内容；在一个高层次上定义一个综合的系统架构（所谓既有深度又有广度的重量级系统），以及明确系统的功能性和非功能性需求。系统架构主要部件的集成通常是通过构建一个有限功能的原型进行阐明。

标志：生命周期构架

(3) *构造*。在构造阶段，注意力从对知识产权（需求和规范）的开发转向对可部署产品的开发。在细化阶段定义的所有架构组件将在该阶段被开发并集成到系统中，且所有功能都将被详细测试。理想情况下，构造阶段被分为一系列迭代，使得每个功能模块在交付时，其输出结果对于自身来说都是有价值的。该阶段结束时，整个系统已被交付（通常被称为 Beta 版本）。

里程碑：初始工作能力

(4) *交付*。交付阶段主要包括将系统交付于用户群体手中进行的一系列活动。这不仅包括软件发布，还包括人员训练以及管理变更。但是，如果在构建阶段或用户测试阶段发现了一些新的问题，或项目出现了一些早期阶段特征相关（通常是需求的变更）的问题，则交付阶段也可能包括下一周期的产品开发以及部署迭代工作。

里程碑：产品发布

请注意，这些阶段并不等同于需求、设计、开发和测试。在 RUP 中，需求、设计、开发和测试会发生在每个阶段的每次迭代中。

在一个使用 RUP 进行管理的项目中，DRAW 是初始阶段的一个必要组成部分，它将帮助我们确定项目范围。记录在自动化范围界定文档中的 DRAW 的讨论

结果，随即可用于帮助规划后续几个阶段。使用 DRA，一个典型的 RUP 架构下的知识自动化项目将包含以下几个高层次结构。

(1) *初始*。针对项目目标达成共识，使用 DRAW 确定决策流程自动化的范围以及高层次的技术需求和非功能性需求，同时使用 DRD 的划分对所有后续阶段和迭代进行规划。

(2) *细化*。基于 DRA 揭示的集成需求，通过建立一个或多个空白的原型决策服务，确定细节性的技术需求以及阐明其与客户体系结构（例如，BPMS）的集成。

(3) *构造*。通过一系列迭代，不断增加原型的功能增量，直到所有决策服务都是完整的。然后为每个增量执行其所需的知识发现和设计任务。

(4) *交付*。执行用户测试、培训和交接等工作，将维护业务知识的责任交还给领域专家。

注意瀑布式方法与 RUP 方法的对比。瀑布式方法中，所有业务知识的发现都是在任何设计工作之前，于需求阶段被收集完成。而在 RUP 中，直到构建阶段——高级架构设计完成且被验证之后——才会执行知识发现任务。每次迭代都将执行知识发现以及详细设计工作，并实施封装了知识的服务组件。

DRA 在项目规划中的应用

DRA 可被用于瀑布式开发方法，或迭代式项目管理体系中，如 RUP。

在瀑布式方法中：

- 若项目启动之前尚未开展 DRAW，则应将其作为需求阶段的首要任务；
- 使用自动化范围界定文档对顶层需求进行定义；
- 在需求阶段，针对 DRD 上的每个知识域分配知识发现任务；
- 基于 DRD，将设计和实施阶段结构化使其能够交付相关功能组件；
- 使用自动化范围文档控制项目范围并协调测试。

在 RUP 中：

- 若项目启动之前尚未开展 DRAW，则应将其作为初始阶段的首要任务；

- ❑ 使用自动化范围界定文档对项目范围进行定义；
- ❑ 对 DRD 进行划分从而定义相应的功能增量，为每个增量执行知识发现、设计、开发和测试任务；
- ❑ 在细化阶段，基于 DRA 揭示的需求，通过建立一个或多个空白的决策服务原型，来阐明服务与客户体系结构（例如 BPMS）的集成；
- ❑ 在构造阶段，逐渐增加原型的功能增量，直到每个决策服务都是完整的。

5.2　DRA在知识发现中的应用

若熟悉业务规则领域，可能你会惊讶，为什么本书到第 5 章才开始讨论知识发现。这样安排是有原因的：知识发现不应该是知识自动化的开端，而应是其结果，在 RUP 中尤其如此。知识自动化项目的范围已通过 DRAW 确立，并使用一个 DRD 进行了刻画。将整个项目范围划分为多个项目增量，并为增量中的每个知识节点分配单独的知识发现任务，就完成了项目规划。通过建立决策服务的空白原型，即可阐明系统技术架构的合理性。现在，你可以逐一对每个决策执行知识发现任务，并扩展决策服务的功能。

在任何一个特定知识发现任务中，应该仅收集业务流程中特定决策点上定义的决策所需的知识。正如我在第 3 章中阐述的，相比试图定义一种适用于所有业务流程和活动的抽象通用业务知识，这个以决策为中心的方法要简单得多。例如，在图 5-1 的增量 1 中，可能你会请求领域专家提供申请风险评分卡以及将风险评分转换为风险级别的一个风险矩阵。这是一个定义良好的工作包，可被表示为两个单独的任务。此时不应该征求任何其他业务知识。

每个知识发现任务的目标，均可从 DRD 及其节点定义中获得，它们声明了知识节点所支持的决策节点，以及这些决策节点的输入和输出。图 5-3 展示了与风险矩阵知识域相关的 DRD 子图。该知识节点是风险级别决策的一个支持节点。在 DRAW 过程中，我们将风险级别决策定义为：使用信用机构数据和申请风险

评分子决策作为输入，并提供一个单独的风险级别作为决策结果（RG1、RG2、RG3、RG4 或 RG5）。因此，在仅仅使用信用机构数据和申请风险评分作为输入时，被收集到的业务知识必须在任何场景下均足以为风险级别决策提供一个决策值，RG1、RG2、RG3、RG4 或 RG5。它不能参考任何其他数据或产生任何其他结果。

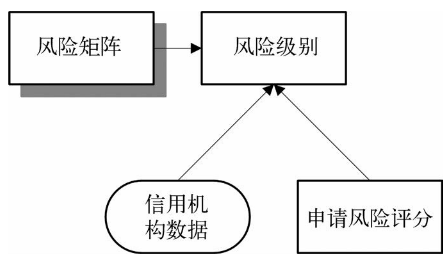


图 5-3　知识发现子图

因此，被划分的 DRD 上展示的信息非常精确地指定了每个项目迭代中将要执行的知识发现任务，同时利用该结构可对任何用于收集知识的工具进行配置。例如，如果使用一个通用的电子表格，可为工作表中的某个标签分配一个知识域，同时使用一个索引标签列出被增量分组的所有知识发现任务，并将其超链接到相关的工作表上。该索引列表还可用于监测项目中知识发现任务的进度。表 5-1 所示为支付能力项目增量中索引标签的可能形式。状态栏追踪了每一个知识域内的知识发现进程，并记录了已由相关领域专家达成共识的被发现的知识。

表5-1　知识发现工作表

增　量	决　策	知识域	持有者	状　态
风险	申请风险评分/风险级别	申请评分卡/风险矩阵	风险/风险	通过/通过
支付能力	每月还款/可支配收入/支付能力	还款计算/支付能力计算/支付规则	风险/风险/风险	通过/完成/正在计算
路径选择	安全性/合格性/申请流通路径	安全性规则/合格性规则/路径选择规则	合格性/市场营销/市场营销	尚未开始/尚未开始/尚未开始

若使用专用知识发现工具，则可更直接地基于 DRD 结构对其进行配置，同时可使用由 BRMS 支持的各种类型知识的模板。

DRA 在知识发现中的应用

知识发现应根据 DRD 中知识节点的情况被分解为不同的发现任务。

在每个知识发现任务中，你只需收集制定特定决策所需的知识。不需要征寻任何其他无关业务知识。

收集到的业务知识必须仅能提供为决策定义的决策值，并只使用 DRD 上与该决策相连的数据，而无需参考任何其他的数据或产生任何其他结果。

业务知识可被收集于一个工作表中，每个知识域使用一个标签进行标记。

知识发现任务的确切性质取决于要收集的知识的形式。正如在第 3 章中讨论的，业务知识可使用业务规则、算法或预测分析模型等形式进行表示。

5.2.1 发现业务规则

目前已有大量关于业务规则发现的文献，包括 Ronald Ross[2] 和 Barbara Von Halle[3] 的开创性作品等，因此我并不想过多地强调这部分内容。本书将着重强调迭代的、以决策为中心的方法能够合理引导和限制业务流程，并且会使许多原有问题消失。规则发现不应是开放式的，它应该被组织成离散的任务，每个任务都有特定的目标：针对每个具体决策确定其业务规则。

这将简化知识发现的过程。业务领域专家们可能会被问到一些非常具体的问题，例如（再次参见图 5-1）：“什么规则可用于判定一个申请是否合格？”他们已经在 DRAW 会议中的高级讨论中探讨过这个决策，并针对这一决策的必要性以及该决策需要什么信息达成了共识，因此规则发现任务亦是对于更为细致的知识域：决策所需的一组完整的业务规则的探索。

正如第 3 章所述，你的目标是定义一个完备且一致的规则集。在收集规则的同时应保证每个规则是原子的、准确的、简洁的和独立的（或陈述性的）。下面对一个良好的规则集所包含的 6 个关键属性进行了简要说明。

(1) 完备性。若一组规则对于所有可能出现的决策场景，均可给出相应决策值，那么它就是完备的。如果已经明确了决策结果的性质，便可通过向专家询问一些

问题来引出相关规则，例如，“决策什么时候会取这个值呢？”特别是当决策用于分类，而你知道其所有可能取值的时候。但这并不足以收集可涵盖所有可能结论的规则。完备性要求规则覆盖所有可能的条件，使得无论规则集被给予什么数据，都至少有一个规则被触发。也许你也会提问：“在这些条件下，规则集将给出什么决策？”

(2) 一致性。若一组规则是互斥的（任何特定情况下只有一个规则被触发），或有一种明确的机制来解决规则冲突（例如提供决策值相关的优先等级），那么这组规则就是一致的。若要检查规则的不一致性，就需要找出具有相同或重叠条件但却输出不同结论的规则。例如，“IF 高度<2.0，THEN 规格=短”与“IF 高度 1.8，THEN 规格=高”在高度范围 1.8～2.0 之间是不一致的，除非在多条规则被触发时同时明确了这些规则的优先级别。

(3) 原子性。若一条规则以最简单的形式进行说明，就称它是原子的。这条规则是一个避免析取（条件之间使用“或”）的有效方法。析取规则“IF a or b，THEN x”应被重新描述为两条独立的原子规则：“IF a，THEN x”和“IF b，THEN x”。这使得规则更易理解，更方便维护和测试。异常的产生通常是因为一个条件需同时检查一个属性的多个取值（例如，“IF month=May，June，July，or Augest”）。

(4) 精确性。每个规则的所有条件都应具备准确阐明领域专家想要表达的知识的能力。例如，在规则“IF 年龄> 25，THEN 状态=成熟”中，我们所说的“> 25”具体是什么意思？是指 25 岁零 3 个月就足够了，还是必须到 26 岁？而更微妙的是，我们要根据什么日期测量年龄，申请日期，当前日期还是服务生效的日期？如果这条规则在多个决策点上于不同的日期被使用，那么它还会一直产生相同的结果吗？

(5) 简洁性。规则应该是简洁的，即可以使用最少的条件用来判别规则覆盖的一组案例）。根据规则 “IF 年龄> 18 岁 and 年龄> 16，THEN x”中的逻辑可知，第一个条件是冗余的。但在“IF 关系=姐妹 and 性别=女”的规则中，我们之所以能够认识到第二个条件是冗余的，是因为我们熟悉“姐妹”的意思。当面对晦涩难懂的业务术语时，许多事情并不总是那么明显。但一个可适用于规则集中所有

规则的类似情况为：若有两个规则“IF 年龄> 16，THEN x” 和 “IF 年龄> 18，THEN x”，那么第二条规则是冗余的。

(6) 独立性。规则应该是互相独立的且评估顺序独立的。若有两个规则“IF 风险 = 500 and 风险<700，THEN 等级=中”。规则集中的规则应该保证在任何顺序下都会产生相同的结果。

一个良好的规则集的属性

规则应该是：

(1) 完备的，足以在任何可能出现的情况下制定决策；

(2) 一致的，规则互斥，或者存在一种明确的冲突解决机制；

(3) 原子的，以最简单的形式阐述规则；

(4) 精确的，准确地阐明领域专家的意图；

(5) 简洁的，使用最少的条件来判别规则覆盖的一组案例；

(6) 独立的，以相互独立且评估顺序独立的方式阐述规则。

若使用隐喻的形式表示规则集，如第 3 章所述的决策表或决策树，则这些属性的实现将会相当简单。在一个决策表中，所有规则的条件都基于相同的属性条目，因此通过对决策表中的列数值进行过滤即可保证规则的完备性和一致性。一个决策树一般是通过逐渐增加用于测试一个或多个条件的节点来构建的，只要节点路径可覆盖所有可能的决策值（例如，通过使用一个“否则”条件），即可保证该决策树是完备的。

根据以决策为中心的方法，我们应该同时检查收集到的规则的范围：

- 规则应该仅用于判定与目标决策相关的结果；
- 规则应该仅使用 DRD 中揭示的决策可用数据域中的数据项。

若事务主题专家提出了一些需判定新决策值或使用新数据域的规则，此时需要针对规则集的扩展范围达成新的共识。

当然，业务规则的条件无需参考 DRD 中展示的数据域，但需与数据域内部特定数据项绑定的业务条款相对应。正如使用特定规则的形式为知识节点提供所需细节描述，也可使用具体业务条款的形式为知识相连的数据节点提供内容。这些条款将在设计活动中（记住，这是一个平行活动，而非下一个阶段）被选出并被用于扩展对象模型，这一过程通常通过添加对象属性来实现。

如果某一数据域被多个决策共享，那么必须首先对该数据域进行交叉检验，以确保重复使用的业务条款指代的是相同的内容，同时不可使用不同条款表示不同规则集中的相同内容。当所有知识域已被完全确定时，被收集并被合理化的这一整套业务条款，将包含决策服务所需输入数据的一个完整规范。

除了一组完整的规则，专家们还被要求为决策服务提供测试案例。和开发者创建的作为项目实施一部分的一组详细的单元测试案例不同，此处的测试案例是少数几个能够反映典型业务情景的特殊案例，这样他们便有信心从决策服务中获得正确的决策值。理想情况下，这些典型案例应该是从历史数据中获取得到的真实案例，并且应该能够涵盖一定范围的决策值。例如，若服务实施分类决策，则至少针对每一类别都应该有一个测试案例。这些典型案例有多种用途：它们有助于阐明分析过程中出现的问题，可被项目实施团队用作进行更详细测试的模板，最终将直接用于用户测试。

务必注意：请记住，我们的目标并不是一定要发现和自动化组织中目前正在使用的所有业务规则，而是要定义一组由领域专家一致同意的、可实施企业最优决策的规则。2005 年，我开始为一家爱尔兰大型健康保险公司构建一个可以支持并自动化医疗索赔评估的系统。在都柏林的第一天，我请一个索赔裁决员带我体验了一个索赔评估的全过程。她从办公桌上一大堆索赔申请表中拿起最上面的一份，然后说道：“嗯，我们首先检查这一部分……”然后，她停了下来，又说道：“嗯，当然，我们也可以只支付这一项。”“为什么？”我问，争取可以收集到我的第一个业务规则。她看着我充满狐疑地说道：“唔，她是个修女，不是吗？”

我很高兴地向大家汇报，这个特殊的规则并未被纳入最终的索赔评估决策服务中。

5.2.2 发现算法

由于算法主要用于表示本身即带有程序性的那些业务知识，因此可以使用伪代码表示。根据我们以决策为中心的方法，可能你会要求业务专家对决策输出结果的计算过程进行定义，该过程可使用一些必要的子过程一路向下计算。此时，还需一如既往地密切留意范围，在该过程中使用的所有数据项都必须来自于 DRD 划分的项目增量内部的数据域，或是由一个子过程从该数据域中推导得出。

算法必须以与规则集完全相同的方式表现完备性：它必须能够为输入数据值的所有可能组合都提供一个决策值。当算法包含基于逻辑条件的分支时，还需应用规则的其他期望性质。例如，这样的条件应当是原子的、精确的、简洁的。当然，算法也有其自身性质，例如，需对循环的边界条件进行正确说明。

但是，已有的关于算法的指南，大部分只是一些优秀的程序设计（一定程度上，只是为其知识发现的目的）。例如，应该一般化某些程序，并尽可能地使其在多个算法中重复使用。此处不对其原因进行详细讨论和说明。

5.2.3 发现分析模型

分析模型并不是从人类专家知识中获取的，而是从经验数据中得到的。这种模型生成的业务知识可被用于自动的制定决策，但同时模型本身也是另外一个决策管理流程——预测分析建模——的输出。预测分析建模以大量历史数据为输入，并生成以模型的形式表现的知识作为输出。基本的预测分析原则在第 3 章中已有解释。

图 5-4 展示了图 5-1 中的申请评分卡知识节点的分析模型和决策自动化之间的关系。预测建模着眼于大量的历史案例，对比分析申请人资料与其最终申请结果之间的关系，从而从申请人资料中找出某些可最优预测输出结果的特征。将这些特征表示于一个申请评分卡中，该评分卡随后可被申请风险评分决策使用，基于当前申请人的资料，预测其申请的可能结果。创建一个预测分析模型可能需要几周甚至几个月的时间，这取决于数据的复杂程度及可用数据项的数量。不过，只

要 DRA 预先确定了模型的需求，凭借精心的规划，建模和实施流程可平行进行。

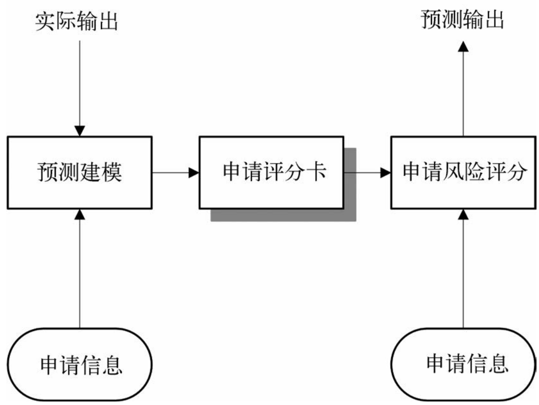


图 5-4　发现分析模型

请注意，若希望使用该模型作为一种有效的预测手段，则决策针对案例所用的判定数据项必须是建模时所用的相同数据项。建模之前，你不可能知道哪些数据项可以最有效地预测决策结果。但是，若你的数据收集流程是固定的，即可通过分析决策点得知：当一个决策第一次在某个决策点上调用预测模型时，哪些数据是可用的。根据这组可用数据，从中选择用于构建模型的特征数据项。另一方面，若该过程十分灵活，可以使用一组更宽泛的数据来建模，寻找最具预测效果的特征，并改变流程从而使得在制定决策时所有数据均是可获得的。有时，会发生需进行协商沟通的情况，例如，市场营销部门可能希望申请表包含更多可以提高客户评分的条目，但是运营部门却希望删除这些条目，以简化新的自动化流程。

最后一点细节：预测模型从来都不是根据所有案例进行构建的，某些特定类型的案例将被排除在外，因为它们是非典型的，可能会干扰预测的输出结果（例如，欺诈案例和破产）。所有这些异常情况都应该在决策服务中被妥善处理。一种常见的解决方法是先对所有案例进行评分，但之后会根据合格性规则对其进行排除。另外一种方法为，排除规则包含于评分卡中，使用特殊的符号值表示异常案例的规则评分。无论采用哪种方法，被评分卡排除的案例通常都会被拒绝。

5.3 DRA在设计中的应用

设计通常是技术架构师的特权，能够反映设计师的关注点和价值观。自下而上的设计往往十分漂亮，因为它能够非常有效地使用计算资源实现其所需的功能。然而，这样的设计有时候可能会令非技术人员难以理解。但是，使业务用户理解他们的知识如何在封装的业务知识中被表达，以及如何将封装的业务知识应用于决策服务，又是至关重要的。毕竟，我们最终期望业务用户来维护这些知识。因此，对于决策服务，一个良好的设计自上而下来看也应该是漂亮的：设计结构应该是十分简洁的，并且能够反映决策制定流程的结构。

自下而上的设计往往针对非常具体的平台，但本书内容试图做到与产品无关，因此在本书中讨论这个问题并不容易。但是，自上而下的设计却可以是纯功能性的。所以，虽然功能表现十分重要，我仍然主要聚焦于设计的结构这部分内容。我的建议是，使用 DRD 展示的决策结构作为一组决策服务、服务组成的决策流以及服务使用的对象模型的设计基础。

5.3.1 在决策点上分配决策服务

默认的假设是，一项决策服务将在业务流程中的一个决策点上被实施，但并非总是如此，因为决策服务可能会被多个决策点重复使用。事实上，正如本书第 2 章所讨论的，决策服务的重用是知识自动化项目的目标之一。此外，也存在单一的决策点在多项决策服务中被执行的情况，此时每项决策服务都是目标决策流程所需的一个组件。在决策点分配决策服务是一种设计判断，而非一种功能性需求。下面是如何进行判断相关的一些准则。

对于业务流程流中的每个决策点，在 DRD 中确定 DRAW 上给出的所有主要决策、所有子决策以及所有支持这些决策的知识节点和数据节点。这个子图定义了将在决策点上调用的决策服务实施的决策的总体范围。对照不同决策点的子图，寻找重叠区域（出现在多个决策点子图中的节点）。

若没有重叠区域，如图 5-5 所示，那么设计判断将十分简单：每个子图可作

为一项单独的决策服务实现，并与相对应的决策点一一映射。

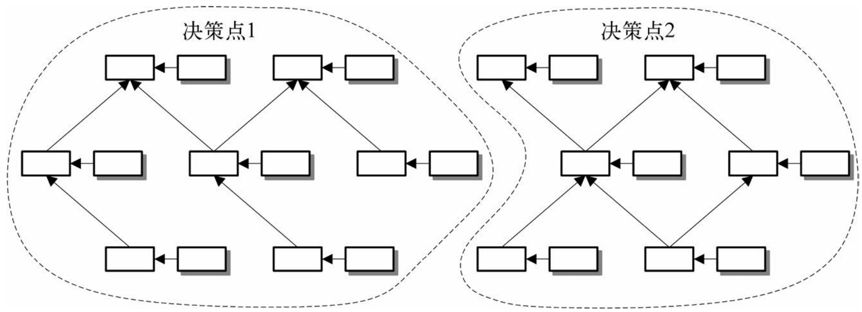


图 5-5　无重叠的决策点子图

若存在小面积的重叠区域，例如一两个决策被分配在了在两个决策点的重叠区域，如图 5-6 所示，这两个决策点仍然应该作为单独的决策服务实施。此时，你有两个设计选项：在第一个决策点的服务结束之后将这些共享决策的结果保存于 BPMS 中，然后以数据的形式提供给第二个决策点；或者对应决策可在两个决策点上独立进行评估。你所选择的设计将取决于许多因素，包括决策结果是否会在不同决策点上发生变化等。

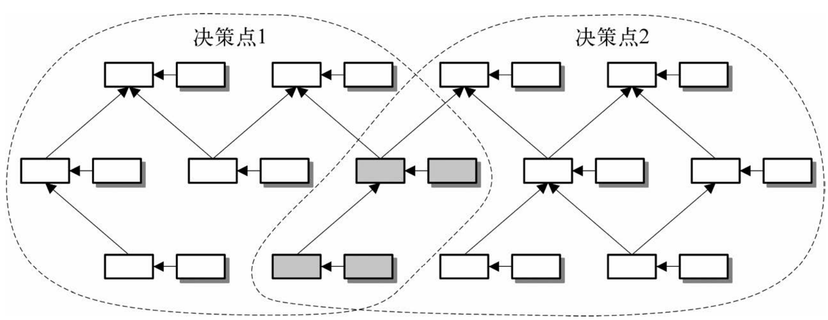


图 5-6　小面积重叠的决策点子图

决策可在多个服务中被评估，但这些决策所用的知识域始终不变。BRMS 为 DRD 维持一个存储其所有业务知识的知识库，并为每一项决策服务部署一个规则子集（见图 3-3）。因此，通过将知识域部署在多项服务中，即可实现知识域的重用。基于 DRD 揭示的决策结构中的知识库的结构和决策服务的设计，可轻松自然

地实现业务知识的重用。

若两个决策点子图出现了大面积的或全部重叠，如图 5-7 所示，那么它们应该在一个决策服务中被实施。两个子图之间的任何细微差别均可通过服务内部对于决策的选择逻辑得以实现。因此，决策服务需要被告知，调用指令出自于哪个决策点，这个决策点标示符应该可以从传递给决策服务的数据中获取。

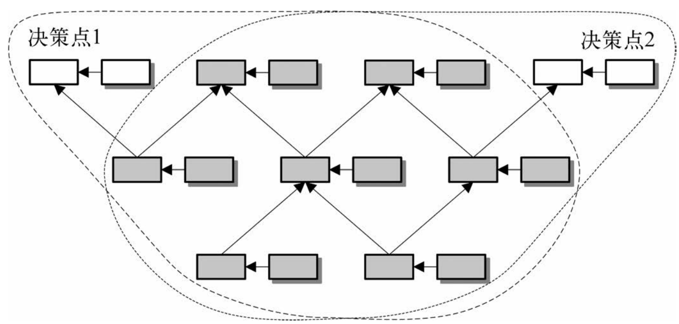


图 5-7　大面积重叠的决策点子图

5.3.2　设计决策流程

许多 BRMS 以及其他实施环境，尤其是那些由正向规则链主导的环境，为详细说明决策的评估顺序提供了一种机制。如果你面向的是这样一个环境，就需将 DRD 中表现的依赖关系网络转化为程序或决策流的形式。该小节中，我将提出一个如何实现上述转化的简单方法。如果你面向的是反向规则链的环境，这可能就是不必要的，DRD 中表示决策之间依赖关系的箭头即可直接被用作规则链。

现在，我们已经为每个决策服务画好了子图，并去掉了除决策节点之外的所有节点。节点之间的箭头指定了服务即将展开评估的决策的局部顺序。通过删除冗余的依赖性，这个简化的网络可以进一步被简化：当两个决策是直接关联或间接关联时，删除直接关联。图 5-8 中所示的局部顺序即是使用这种方法从图 5-1 中的示例 DRD 中得出的。

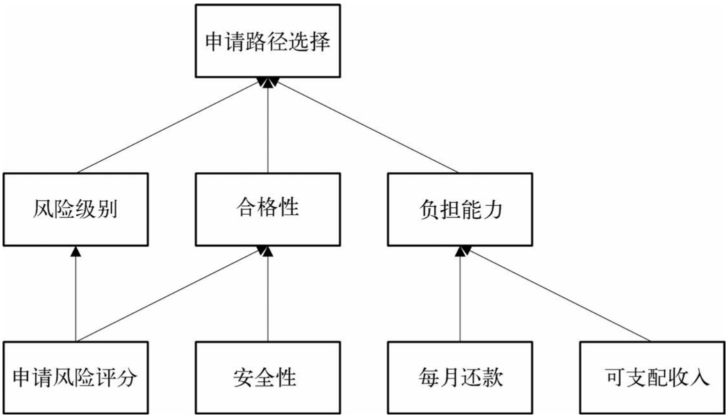


图 5-8　决策的局部顺序

在某些 BRMS 中这个局部顺序可能足以支撑一个决策流，但某些流程则要求你把它简化为一个不包含并行任务的线性流程。可以通过为任何并行节点简单地选择一个任意的序列来实现局部顺序的线性化（一个局部顺序有很多兼容的可能序列）。如图 5-9 所示，其结果是一个实施决策点子图的服务的决策流。流程中的每个任务，都必须通过使用相关的业务知识触发 BRMS 展开决策评估。

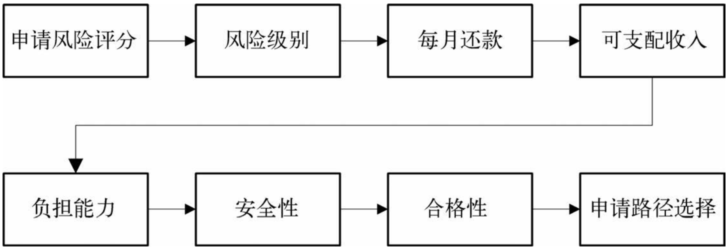


图 5-9　一种可能的决策流图

若遇到以下三种情况，则可能需要扩展这个简单的方法。

(1) 上文方法假设所有数据均在调用时被提供给服务，所以这些数据与决策流是不相关的。这通常是一个典型的无状态决策服务会产生的情况。如果，抛开我在第 2 章的建议而要求服务调用其他系统进行数据采集（例如，查询数据库），那么这些数据节点也应被包含于顺序分析中。

(2) 如果你在功能并未完全重叠的多个决策点上重用一个决策服务（如图 5-7 所示），则可能出现某些决策位于决策点范围之外的情况。此时，需借助条件性子流程的分支对图 5-9 的线性序列进行阐明，以确保每个决策点仅对其相关决策展开评估。

(3) 最后，若某一决策是为多个对象实例制定的（例如，选择并配置一个可能的供给贷款列表），则你的决策流可能还需包含循环，为列表中的每个实例评估其必要决策。

假如决策按照某一序列被逐一评估，且该序列与 DRD 中隐含的局部顺序兼容，即可保证，当每个决策被决策服务评估时，均可获得其所需要的全部信息（输入数据及子决策结果）。

5.3.3 设计对象模型

通过知识发现，我们明确了规则、算法和模型中提到的所有业务条款，并将其与 DRD 中的节点关联了起来。输入条款与数据节点相关联，输出项是决策节点的具体决策值。现在你面临的任务是设计一个对象模型，并将所有业务条款映射其中。

如第 3 章所述，当需要将知识表示为业务规则时，对象模型的设计至关重要。对于对象模型，一个好的设计有两个目标：一个目标是向上的，朝着业务领域；另一个是向下的，着眼于技术实现。

(1) 对象模型的结构应是合乎逻辑的：它能够反映知识域的逻辑结构，表示并涵盖可在业务领域中自然讨论的事物对象，并且能够描述这些事物之间的关系。当对象模型可准确地反映业务领域时，领域专家将更易表示规则，分析师更易理解规则，且规则的实现更简单、执行更高效、维护也更方便。当作为“事实模型”时，对象模型的重要性更是显而易见：任何没有在对象模型中表现出的知识都必须清晰地包含在规则中。

(2) 对象模型的结构应是基于事件的：它应该能够反映产生和消耗数据的流程

步骤。数据表现为业务流程中（例如，当一个界面被提交或者当数据库被查时）特定点上的可用数据包，并以数据包的形式被使用（例如，当把数据传递给一个决策服务或显示在一个界面上时）。这些数据包是业务流程中数据重用的自然单元。此外，一个典型决策服务调用的响应时间很大程度上是由数据接口的效率决定的，因此一个服务的对象模型应该仅描述这些可用的和需被使用的数据项。

这两个设计目标必须在考虑范围之内，但它们之间往往存在一些冲突，在某种程度上，有时很难在一个模型同时满足这两个目标。特别是当有多个不同的对象，且业务流程中多个点上使用的一个对象（例如申请人）拥有多个不同属性时，极易出现冲突问题。当冲突发生时，该如何权衡最佳选择？是否会存在一种妥协？

可以毫不犹豫地说，在对象模型的输入侧（数据被传递到决策服务），逻辑结构远比事件结构重要得多。对于重用和效率问题，存在许多纯粹的技术解决方案，它们主要取决于解决方案架构，但对其内容的讨论远远超出了本书的范围。不过对于业务领域而言，对象模型的真实性对于封装了业务知识的决策服务目标至关重要，在这一点上绝不能妥协。

带有预见性的自顶向下的原则在此处十分有用。当主持 DRAW 时，聪明的决策分析师会考虑对象模型的可能结构，并通过对源于同一事件或单一对象模型相关的数据进行划分，尝试对 DRD 数据域进行定义。随后设计师会发现，在一个逻辑合理且有效的对象模型中对这些进行安排会更加容易。

在输出侧（即服务返回结果），逻辑结构往往并不会那么清楚。尽管输出结果关注领域内的对象，但它们是在特定的决策点上制定的特定决策结果，在被使用，尤其是被后续决策点所使用时，具有至关重要的相关性。一个对象（例如个人风险评估）的单一逻辑属性可能会通过多个决策点返回，其后续决策点可能需要区分这些返回结果，甚至可能需要比较各个返回结果的值。出于这个原因，我通常喜欢根据决策点，其次是对象或决策，来构建对象模型的输出侧。

图 5-10 展示了图 4-1 中的示例 DRD 的一个对象模型的骨架，由 XML 模式定义语言（XML Schema Definition，XSD）定义。任何真正的对象模型都远比该模

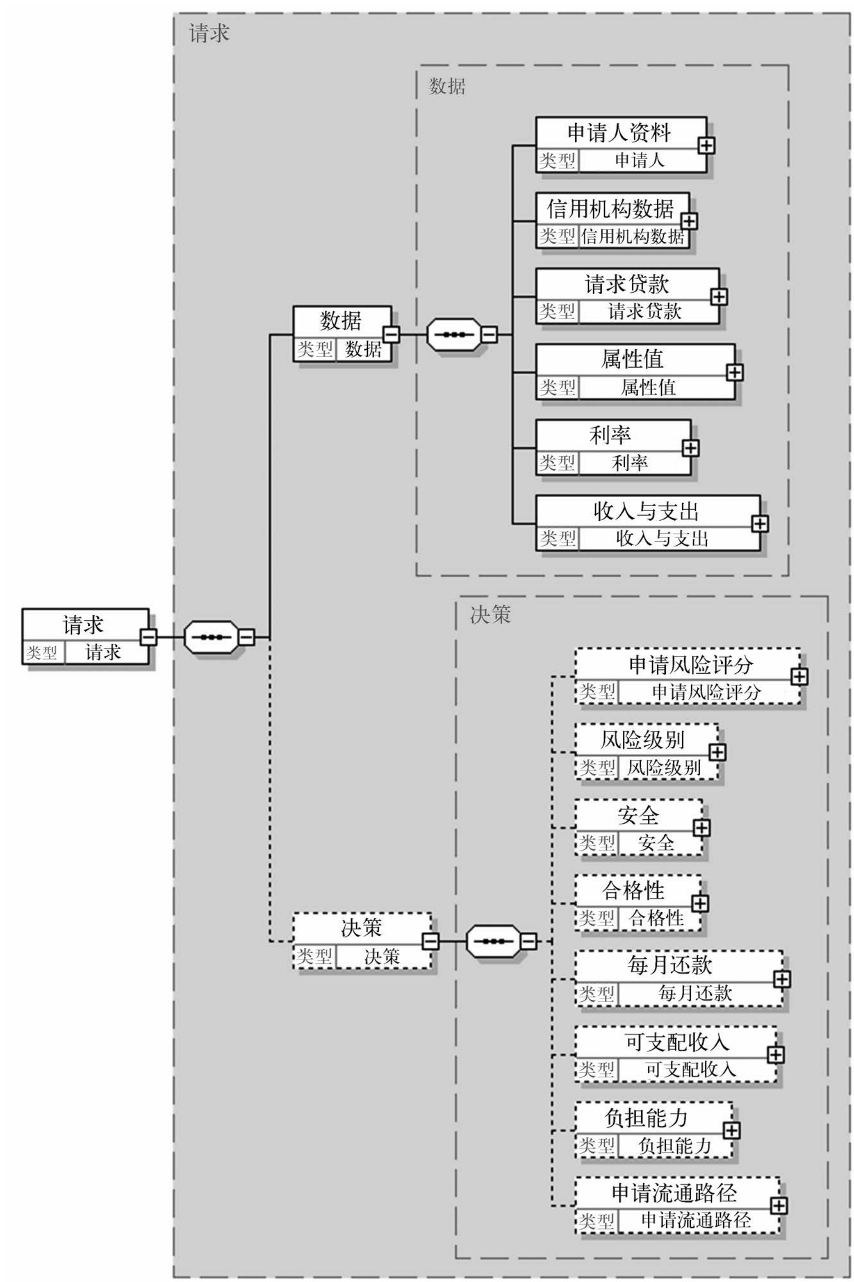

请求
数据
申请人资料
类型 申请人
信用机构数据
类型 信用机构数据
请求贷款
类型 请求贷款
数据
类型 数据
属性值
类型 属性值
利率
类型 利率
收入与支出
类型 收入与支出
请求
类型 请求
决策
申请风险评分
类型 申请风险评分
风险级别
类型 风险级别
安全
类型 安全
合格性
类型 合格性
决策
类型 决策
每月还款
类型 每月还款
可支配收入
类型 可支配收入
负担能力
类型 负担能力
申请流通路径
类型 申请流通路径

图 5-10 一个对象模型

型更复杂，但是此图展示了一个对象的基本架构。图中有两个主要对象：数据和决策。DRD 中的所有数据域均被表示为数据对象的属性，而 DRD 中的所有决策均被表示为决策对象的属性。数据的每个属性本身也是一个包含多个属性的对象，子对象的每个属性表示数据域中的一个数据项，例如，申请人资料对象中会包括一个描述申请人的每一项数据的属性。类似地，决策对象的每个属性也是一个包含多个属性的对象，子对象的每一个属性对应决策结果中的一个数据项。

图 5-10 所示的模型定义了一个顶层请求对象。当决策服务被调用时，它将接收一条包含此对象实例（一个请求）的信息，这个实例的所有数据属性均使用案例数据填充。当决策服务实施决策时，该服务将使用已获取的信息填充决策属性，然后在完成时返回一条信息，该条信息将包含完整的结构：数据和决策。正因如此，模型必须包含数据属性，而决策属性则是可选的。（这仅仅是一种在决策服务中传递消息的可能方式，另一种则需要两条独立的消息，一条用于请求，另一条用于响应。）

基于其他决策结果进行决策制定时，决策服务将参照之前在决策对象中填充的数值。这使得业务逻辑的依赖性一目了然。当决策的结果值被存放于外部且需被传递给后续决策点时，它们应该以完全相同的结构（即决策对象而非数据对象）呈现给后续决策点。一个安全的（但有时代价高昂）方法为，将每个决策点的所有决策结果全部呈现给其所有的后续决策点。

除了对象模型的结构设计，还需对模型之中应该表现哪些数据的问题进行讨论。在输入侧，存在两种极端情况。

(1) 仅包含决策制定时所需使用的数据。这将最小化对象模型的规模，从而减少需传递给决策服务的消息，这是决定其性能的最重要因素之一。

(2) 包含决策服务被调用时的所有可用数据。这将最大化解决方案的敏捷性，因为新的规则可在任何时候通过使用知识维护界面进行添加，而不需要对软件进行数据接口扩展及修改。

通常，解决方案会介于两者之间，是某种形式的折中，主要包含那些可能在规则中使用的对象模型中的数据项。当然，模型永远都不应该包含那些不会在决

策中使用的数据项。我曾经与一个客户有过一次艰难的讨论，起初他希望把描述申请人的所有数据都置入模型之中，最终他让步了，但仅仅是由于我指出，无论出于什么目的，在决策机制中使用某些数据项事实上都是违法的，所以它不应该被纳入业务规则并提供给用户。

在输出侧，也存在两种极端。

(1) 仅包含 DRAW 确定的主要决策结果。这将在提供业务流程中服务所需的决策的同时，最小化对象模型的规模。

(2) 包含由决策服务判定的所有结果，包括所有子决策的结果及其推导过程中使用的中间决策值。这将允许对于整个决策过程的审计，并且这对在测试期间进行调试尤其有用。

再次强调，这通常需要一个折中方案。我通常主张返回 DRD 上所有决策的结果，但没有必要返回它们的内部及中间决策值，除非确定这些值对于业务用户具有十分重要的意义。

DRA 在设计中的应用

基于 DRD 中揭露的决策结构进行项目结构设计。

通过确定每个决策点的 DRD 子图及其重叠面积，在决策点上分配决策服务。

在使用反向规则链的 BRMS 中，使用 DRD 中节点之间的依赖关系作为链规则；在使用正向规则链的 BRMS 中，基于规则链寻找决策的局部顺序并对其线性化，以产生一个线性的决策流。

可能的话，设计一个对象模型，使之既能够体现领域知识的结构，又能够体现产生或消耗数据的事件的结构。

5.4 DRA 在实施中的应用

如图 3-3 所示，BRMS 的核心部件是一个知识库，用于存储部署决策服务所

需的业务知识。围绕知识库，有一系列的工具和设施，通常包括：

- ❑ 知识库结构的配置设施
- ❑ 库内业务逻辑的直接编辑设施
- ❑ 业务用户界面的创建设施，它允许主题事务专家在库中定义和维护自身知识
- ❑ 预测分析模型的导入和编辑设施
- ❑ 一个可执行业务知识的业务规则引擎
- ❑ 封装了库中知识的决策服务的开发工具
- ❑ 决策服务的测试和部署工具

此处，SME 的作用是提供和维护业务知识，架构师和研发团队的作用是装配所有的支持性基础设施：如创建和配置知识库，为 SME 的知识导入提供支持设施，以及研发决策服务的部署方式。

5.4.1　基础设施开发

第一项任务是安装 BRMS 产品，并搭建其规则开发、测试及产生的环境。随后，架构师可创建知识库，设置部署机制，并根据任何接口的特定需求展开研发。一旦架构师定义了决策点与决策服务之间的映射关系，即可为集成测试创建一整套决策服务的空白原型。若使用 Web 服务，就不必在该阶段定义完整的对象模型，因为 WSDL（Web 服务描述语言）简单地将数据接口指定为一个字符串。一个拥有两个输入数据域和一个单一决策的高级结构，即可实现该目标。若项目恰恰在 RUP 下运行，则这些工作都应该在细化阶段完成。

随后团队可按照已有的设计为每个服务创建一个决策流，并为流中的每个决策任务定义一种“实施类型”。实施类型或者是一个标准类型（如一个简单的规则集、隐喻或功能），或者是针对某一特定项目的定制类型。实施类型定义了：

- ❑ 知识的形式
- ❑ 知识维护界面的模板

- 一种带有可重用代码的标准实施方法
- 对象模型中包含的可重用的对象
- 一种测试的标准方法

若决策流中决策任务所需的功能在绝大程度上都可由标准实施类型满足，那么这将是项目实施的一个巨大优势。这是决策分析师的部分责任所在，如第 4 章所述，分析师将使用一种带有预见性的自顶向下的方式来分解需求。某些 BRMS 产品仅能执行某些特定类型的知识，例如，某些产品主要基于决策表，某些主要基于规则。因此，标准实施类型应符合实施决策所用的 BRMS 的性能，同时，决策分析师在引导 DRAW 时应该了解 BRMS 包含的标准实施类型。

该指令系统将逐步发展壮大。研发团队将从 BRMS 中选取几个带有标准实施方法的标准实施类型，例如，一个算法类型将作为一个函数调用来实现，一个业务规则类型将作为一个单一规则集的应用以及一些支持的隐喻来实现。当团队实施连续的项目时，BRMS 应该建立一个自定义实施类型的库，将那些可能会被重用的库进行通用化处理，并添加到标准类型中。随着时间的推移，组织会针对自身独特的决策风格，开发一个标准执行模式的工具箱，这将使得设计重用和代码重用成为可能。这也将使得业务领域专家更加熟悉标准的知识形式，并更好地理解何时需使用哪种知识形式。

一些适合纳入标准实施类型库的常见决策模式，将会在第 6 章中进行讨论。

5.4.2 知识配置

正如第 1 章所讨论的，衡量决策收益（实施决策管理的好处）的一个重要维度为：能否在不断变化的经营策略中提高决策敏捷性。当策略在 BRMS 中被自动化为业务规则后，敏捷性取决于这些规则更新及重新部署的速度。无论你对 BRMS 的知识编辑设施有多么熟练，若使用传统的包含需求改变、技术参数、实施和测试的 IT 开发周期，将会严重影响经营决策的敏捷性。因此重要的是将维护工作分为两层，从而使得某些受约束的知识维护活动可由业务人员而非 IT 人员安全执行。

通常 BRMS 会提供一个可被 SME 用于创建单独知识维护界面的设施。这种接口针对某一领域中知识的形式和范围量身定制，并允许 SME 仅对知识进行编辑，而非编辑执行知识的基础设施——这是 IT 开发团队的任务。这个接口利用了可执行知识的本质：知识逐渐成为其执行系统的数据状态。因此，可使用类似于保护数据库数据的控制（授权级别、口令保护等），而非那些类似于软件的版本发布的控制，来更新知识。

应当根据知识域维护的业务功能，对接口设施进行分组访问，如图 5-11 所示。基于图 4-1 中的示例 DRD，而根据表 5-1 中定义的拥有者列的责任，进行知识域维护的分组。对执行这些知识域的决策基础设施进行维护，仍然是开发团队的责任。

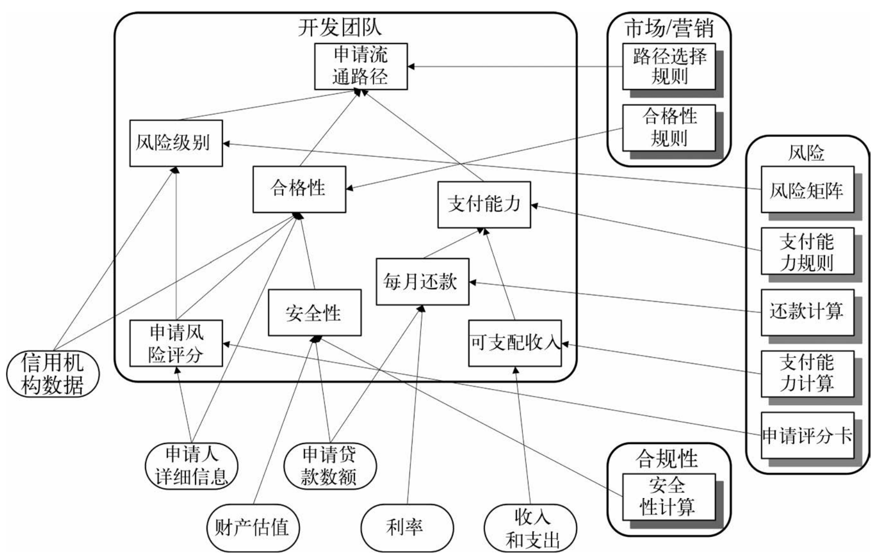


图 5-11　知识维护界面

相比于一个完整的开发环境，知识维护界面更易被使用，因为知识可使用有意限制的模板进行编辑，从而简单地使用户避免语法错误。通过识别一组由业务领域专家维护的、离散的知识域，DRA 允许非常具体地设计这些模板。同时 DRA 可判别：实施某一知识决策还需哪些数据，以及这些数据可在何时可用。若知识维护模板是受限的，那么被规则包含的数据项将仅仅是那些在决策评估时可用的

数据，此时亦可排除由数据丢失所造成的运行错误。

业务知识的初始配置可以由开发团队或 SME 设定，无论由谁设定，都应该仅使用知识维护界面而非完整的开发环境。这就为确保模板是适当且方便的提供了一个有益的检查。一个良好的方法为：知识由事务主题专家配置，由开发团队提供支持。这使得在 SME 可以在开发团队交接知识维护界面之前就能够熟悉界面的使用。

5.4.3 测试

任何软件实施项目在投入运行之前，都必须进行 3 个级别的测试，以阐明该决策服务可提供所需的功能，并可安全地部署在操作环境中：集成测试，单元测试，用户测试。

1. 集成测试

集成测试阐明，项目的体系结构是合理的，即，任一决策服务均可被客户端调用并返回结果，无需考虑决策服务的决策制定功能。这种测试有时被称为 Hello World 测试，它往往取决于开发者的习惯，可能仅包含在服务中写入一行代码并返回一个简单的响应这样的功能。

RUP 中要求使用集成测试实现细化阶段的生命周期架构里程碑。如早前所述，可通过为每个决策点投放一个空决策服务来实现这一目的。不同项目中空服务集的空的程度不同。若服务不包含任何功能，即可以展示其运行架构（即服务调用及响应），但不可展示任何知识维护设施。如果希望将服务功能包含在集成测试的范围之内，无论通过知识维护界面进行多么简单、可配置的操作，服务都需要包含一些决策制定相关的实质内容。

在最简单的情况下，这可以包括一条使用单一输入数据项并返回一个单一决策值的规则（IF 年龄<18，THEN 流通路径=拒绝）。知识维护界面只需允许改变规则中的年龄限制即可。然后，可通过以下操作阐明该服务架构：使用知识维护界面改变年龄限制，使用不同年龄的申请人案例调用决策服务，然后观察他们在

当前配置的年龄限制条件下会被接受还是会被拒绝。

2. 单元测试

单元测试阐明，决策服务中的每个组件都可单独运行成功。即无需考虑这一决策组件与其余架构组件的集成，便可制定正确的决策。

可参考在知识发现工具中定义的知识，对决策的正确性进行检查。每个业务规则都有其特殊情况，此处我们主要阐明每一条单独的规则都可被正确执行，而不是试图对输入数据的每一种可能组合进行测试，这在大多数情况下是不切实际的。相反，我们希望能为每个规则定义一组最小测试集，目的为训练规则的主要行为模式。一般而言，有如下这些测试。

- 对于合取（使用 and 连接条件）：
  - 一个正面测试案例，当满足规则的所有条件时，触发规则
  - 规则中每包含一个条件即为其设定一个负面测试案例，使规则因不满足某一条件而不能被触发
- 对于析取（使用 or 连接条件）：
  - 一个负面测试案例，当所有条件均不被满足时，不能触发规则
  - 规则中每包含一个条件即为其设定一个正面测试案例，使规则因满足某一条件而被触发

如之前章节所述，在规则中包含析取并不是一种最优的做法，但是，当使用一个条件测试某个属性对应的可能取值列表时，析取即为其最优选择（例如，就业状况=学生 or 失业 or 私营个体）。

注意，这种方法使得带有 n 个布尔条件的合取规则需进行（n+1）次测试，而非 n^2 次测试。因此，例如，若一条规则的条件“IF a and b and c， THEN...”，那么其测试案例包括：

- a 为真，a 为真，c 为真，规则被触发；
- a 为假，a 为真，c 为真，规则不被触发；

- ❑ a 为真，a 为假，c 为真，规则不被触发；
- ❑ a 为真，a 为真，c 为假，规则不被触发。

同样地，若一条规则的条件为“IF a or b or c，THEN...”，那么其测试案例将包括：

- ❑ a 为假，b 假，c 为假，规则不被触发；
- ❑ a 为真，b 假，c 为假，规则被触发；
- ❑ a 为假，b 真，c 为假，规则被触发；
- ❑ a 为假，b 假，c 为真，规则被触发。

一个简单的方法是使用一个标准的通用案例作为标准测试集，然后，每个测试案例将阐明：

- ❑ 对标准测试集中的案例进行一处或多处的微小改动，即可创造出期望的条件组合
- ❑ 期望的结果——执行规则后产生的决策值

单元测试包含大量此类测试，因此需要使用一个自动的测试设施对所有定义测试调用决策组件，并对实际运行结果和预期结果进行检查对比。BRMS 通常会提供这样一个工具。

3. 用户测试

用户测试阐明，整个系统可像终端用户看到的那样正确运行。此时需考虑系统集成和决策机制，但不需要完全覆盖所有可能的案例。这可能需要包括端到端的测试，以及使用完备的数据集从 BPMS 中逐一调用所有的决策服务等内容。

相比于单元测试，用户测试是典型的基于更小的测试集，但是要求测试案例必须能够表现真实案例特征，并要求完全覆盖业务流程流中所有可能的流程。因此，选择的测试案例应该覆盖 DRAW 的第 2 阶段上定义的主要决策的所有可能结果（参见第 4 章）。所以，图 4-5 中示例所需的测试案例可能包括：

- ❑ 使用调用信用机构之前所拥有的数据即可拒绝申请的案例
- ❑ 必须调用信用机构数据的案例
- ❑ 无需调用信用机构数据的案例
- ❑ 使用信用机构数据即可拒绝申请的案例
- ❑ 待定需评审的案例
- ❑ 原则上合格但没有发放贷款的案例
- ❑ 原则上合格且发放了贷款的案例

每个测试案例都包括输入数据以及期望输出结果两部分内容，它们必须由企业定义，从而确保测试案例与真实情况保持一致。注意，由于测试案例基于 DRAW 的结果，企业可在项目初期即对测试案例进行准备。而准备的案例随后可作为知识发现、设计和实施的资源，并可被开发团队用作单元测试的测试案例的基础。使用测试案例作为项目的开端，而非将其留到项目最后的这种方法，是软件开发中敏捷方法的一个特质。

DRA 在实施中的应用

决策的实施主要基于标准实施类型的一个库，每个类型定义了：

- ❑ 知识的形式
- ❑ 知识维护界面的模板
- ❑ 使用可重用代码的标准实施方法
- ❑ 对象模型包含的可重用对象
- ❑ 测试的标准方法

标准实施类型将根据特定项目的定制类型进行填充。随着时间的推移，可被重用的定制类型将被通用化处理并被添加到标准类型库中。

提供针对每个知识域的专门的知识维护界面，仅允许事务主题专家编辑知识，而非编辑基础设施。

根据知识的拥有者来对知识维护界面进行分组维护。

针对基于大量综合案例的单元测试，检查每条规则的每个条件。

针对基于少量真实案例的用户测试，选择可覆盖 DRAW 中定义的主要决策的所有结果的测试案例。

5.5 知识生产线

基于知识的系统历来被视为一种复杂的手工引擎，它要求技术精湛且经验丰富的知识工程师与之配合。但是使用正确的方法可减少引擎所需的专业技能。将决策服务构思为可适用于一个定制框架的许多相似部件，即是一种正确的方法。有时需要经验丰富的工程师构建这样一个框架，但需要大规模生产技术来制造这些部件。正是这种洞察力，使得亨利 · 福特在 20 世纪初即可极大地降低汽车的生产成本。今天，我们也能够在决策自动化中获得类似的好处。

在一个知识自动化项目中，实际上你是在创建两样东西：

(1) 一个制定决策的机器

(2) 一条制造该类型机器的生产线

DRA 和 RUP（或任何其他可交付的迭代方法）的结合，使得创建一条发现、设计和实施知识的高效率生产线成为可能。构建阶段的每一次迭代均对规定的业务知识增量进行收集、分析、编码并将知识封装在决策服务中。这涉及对知识在若干阶段中进行传递的过程，如表 5-2 所示。

本章对迭代包含的所有阶段进行了讨论。完成所有的单次迭代工作即需持续几周，这是一个非常艰难的任务，除非可保证以下几点：

- 一个良好的需求分析，使用 DRAW，简洁地定义迭代实施的项目增量
- 定期迭代，循序渐进的工作
- 保证 SME、分析师、架构师以及开发团队等所有参与人员之间都有非常良好的工作关系

❑ 保证项目质量及其进度

当所有上述保证都到位时，其成效是非常显著的。它可以令项目工作变成一种享受。当然，要实现这些成功的条件，项目赞助者以及项目经理责无旁贷。

表5-2　知识生产线中的阶段

阶　段	任　务	描　述	工作人员
1	知识发现	发现并分析业务知识	决策分析师和事务主题专家
2	设计	设计所有新组件（服务、流、任务、知识维护接口	架构师
3	开发	实现新组件的所有新组件	开发人员
4	知识配置	配置业务知识	开发人员或SME
5	单元测试	测试可交付组件的表现	所有人员

生产线的另外一个十分重要的优势在于其可追溯性。使用 DRA，将增量中的知识生产单元定义为 DRD 上的决策和知识域的集合,可允许对于知识生产线的全程追溯：从可交付组件，到设计和知识发现，一直到自动化范围文档中的原始需求。例如：

❑ 每个交付的决策服务负责实施 DRD 上的某些主要决策,决策在业务流程流中的一个或多个决策点上被制定；
❑ 每个交付的决策流负责实施一个 DRD 子图；
❑ 决策流中每个交付的决策制定任务负责实施 DRD 上的一个决策节点；
❑ 每个交付的知识组件（如规则集或分析模型）负责实施知识发现任务中的某一单一知识领域，同时还提供了 DRD 上一个知识节点的详细需求。

这种程度的可追溯性在一般软件开发中十分少见，在基于规则的系统中尤其如此。正如本章开头所述，这是使用 DRD 中揭示的决策制定流程结构来组织和定义知识生产线上所有任务产生的结果。

5.6 参考文献

1. Rational, “Rational Unified Process：Best Practices for Software Development Teams,” Rational Software White Paper TP026B, rev. 11/01, 2001.
2. R. G. Ross, *Principles of Business Rule Approach* (Boston： Addison-Wesley Profes- sional, 2003).
3. B. Von Halle, *Business Rules Applied： Building Better Systems Using the Business Rule Approach* (Hoboken, NJ： John Wiley & Sons, 2001).

第 6 章

常见决策模式

经历了从顶层业务需求分析到底层决策服务实现这样一个自顶向下的旅程，终于到达了本书的最后一章。本章包含了之前描述的一些概念、方法及技术的随机组合，目的是展示在某些特定情景下如何实现决策机制自动化，为之前章节中讨论的相对抽象的概念提供一些应用的实例，并为刚接触 BRMS 的读者提供一些学习窍门。首先，我将介绍决策自动化的一些常见类型的示例，然后将围绕协同决策机制实现的多种方式展开讨论，包括人与决策服务的协作，并以此结束本章内容。

6.1 决策实施类型

经验被认为是一种不可取代的资本。实施过一些知识自动化项目后，你会发现在项目中经常遇到许多仅有些许差异的、类似的决策模式。有效识别常用决策模式可极大地减少流程分析阶段所需花费的时间，同时允许使用一种专门的用于此类项目开发的标准实施类型的工具包，正如我们在第 5 章所讨论的。最简单的类型是一种被称为“基元”的决策模式，它仅包含一个标准格式的知识域。基元包括第 3 章所描述的规则集及其隐喻、算法和分析模型等。显然所有这些都应该以标准实施类型的形式提供。不仅如此，工具包还应该反映某个组织决策机制的独特风格。

本章展示了一些我经常遇到的非基元的决策模式，它们或许可作为通用标准

实施类型工具包的一个组件。以下内容并未展示所有常用决策模式，仅仅是一小部分有用的示例。

6.1.1 带理由的分类决策

1. 用法

一般而言，制定一个基于规则的分类决策是一件非常简单的事情：在可选类别的离散集合中选择其中一个类别。此类决策存在于很多场景中，通常用于选择流通路径。BPMS 系统通过使用其决策结果（一般为“接受”“待定”或“拒绝”），来决定案例在流程流中经过的节点序列。图 2-8 中所示发放流程模板包括三个此类决策，每个决策点上有一个决策。

然而，直通式流程的异常处理不仅需要主要决策的值，还需解释制定决策的原因，从而为用户提供相应的申请指导及评审追踪。一种简单且便捷的解释方式是为向用户提供一个理由列表。这些理由通常被用于评审任务管理，评审员必须对列出的待定原因进行逐一审查，并将可通过的申请标记为“接受”。注意，通过使用决策服务输出的待定信息，评审活动可由 BPMS 代为执行，评审过后的案例无需再使用 BRMS 的决策服务对其评审结果进行评价。

通过为某一决策设定决策默认值并使用特定规则描述该决策的异常情况，即可保证其规则集的完备性。通常情况下，决策的默认值为“接受”或“继续”，并设定一组描述待定或拒绝案例的规则。该方法对于某些偏向于使用“拒绝”作为决策默认值并设定一组“接受”条件的用户而言，可能有些违反常规。但是，略加思考之后你就会明白为什么后者并不是一个好方法。

假设判断合格性需要 20 个判定条件，使用以下条件作为开端：

- ❑ 年龄>= 18
- ❑ 居住地 = 本地
- ❑ 没有破产

此时，不能将这些规则分开考虑，因为显然“年龄大于 18 的申请人的申请均可被接受”或“本地居民的申请均可被接受”都是不正确的。“接受”决策的规则集必须由包含 20 个判定条件的单条规则组成。

- IF 年龄>=18 and 居住地=本地 and 没有破产 and [其他 17 个判定条件] THEN 流通路径=“接受”

这样的规则很难维护。因为任一条件不被满足，规则都不会被触发，此时我们也无法获得申请不被接受的理由。而另一种方法，通过将决策默认值设定为“接受”，可进行如下判断：

- IF 年龄<18 THEN 路径 = “拒绝”
- IF 居住地 != 本地 THEN 路径 = “拒绝”
- IF 破产 THEN 路径= “拒绝”
- 其他 17 条判定规则

这就给出了 20 条独立的规则，我们可使用一个知识维护界面对其进行编辑和维护。

通常情况下，规则集中存在一些带有两个及以上类别选项的规则（例如“待定”和“拒绝”）。正如我在第 3 章中所解释的，我们需要为类别而非规则的选择，定义一个与顺序无关的优先级别。因此，每条规则均可被认为是在选择一个类别，最终输出结果为被触发的任意一条规则对应的具有最高优先级别的类别。

为每条规则绑定一个理由，则同时可在规则触发时提供一条相应的评审记录，记录可能包含以下信息：

- 对于一个待定类别值，记录该待定案例应随后被提交于哪个评审团队
- 对于一个拒接类别值，记录该值可否随后被改写

决策服务应该返回最终决策结果，附加一张记录了所有被触发过的规则的列表。

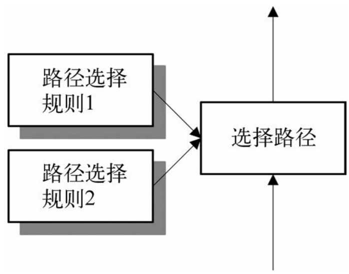


图 6-1　带理由的分类决策模式的 DRD 组件

2. 模式

图 6-1 和图 6-2 展示了带理由的分类决策模式。

- DRD
  - ❑ 包含一个单一决策节点以及一个或多个知识节点，每个知识节点都持有关于某一业务规则域的一个规则集，如图 6-1 所示。决策的输出结果为：一个路径类别及其相应的理由列表。
- 知识
  - ❑ 包括一组决策值，其中一个决策值被定义为默认值，同时为不同冲突解决方案确定优先级别（如第 2 章所述）。
  - ❑ 包括一个或多个规则集，每个规则集描述默认决策值的一种异常情况，包含以“IF <条件> THEN 提议(<类别>，<理由>)”形式表现的规则。除了抽象模型外的所有规则都应该使用相同的规则提议函数，这允许我们使用相同的决策模式对多个规则集及其组合进行处理。
- 对象模型
  - ❑ 一个带有类别属性及其理由列表 x 的决策对象。
  - ❑ 一个理由对象，记录了一条规则提出的类别决策、相关决策理由以及其他必要属性。

图 6-2 展示了一个流通路径选择决策对象模型的示例，其类别为一条流通路

径。注意到，流通路径对象可在其他类似决策中被重用。

- 决策基础结构
  - 包含决策流中的一个独立决策任务，其初始决策值为默认类别值，该任务随后可在相关知识域中使用其所有决策规则集。
  - 包含一个规则提议函数，用于实现策略规则任务。该函数可向理由列表中增加理由，并可在提议类别比当前类别优先级高时，更新决策的输出类别。

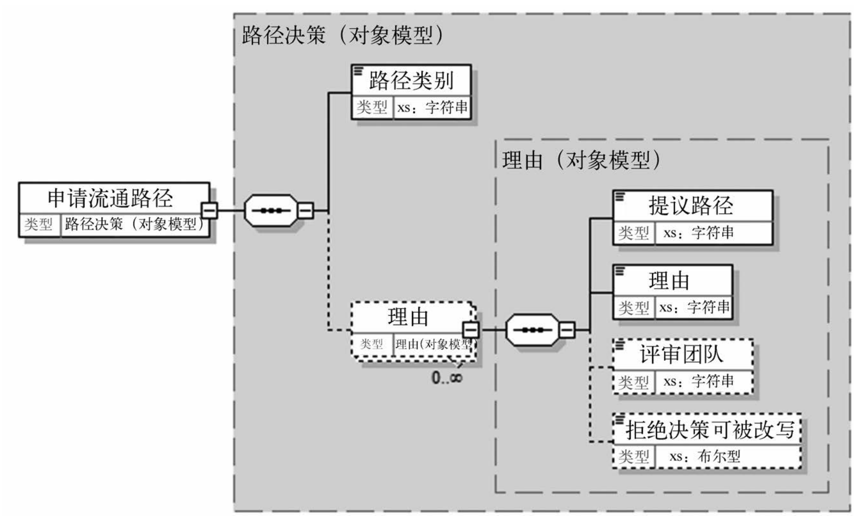


图 6-2　对象模型类别模式的原因

6.1.2　带异常记录的规则决策

1. 用法

作为一个分析师，如果你在与主题事务专家确认一个业务规则集是否完全定义了组织的政策，得到的回应却是“是的，除了……”时，一切将变得有趣起来。

异常很难被建模，通常它们都十分具体，并且可能对业务流程有特殊含义。例如，假定我们有一个通用的政策规则：特定类型的支付方式(x)是不合格的，除非它以某种特定方式(y)被授权。它可以被表述成类似于这样的形式：“IF 支付类

型 = x ，and not 授权类型 = y THEN 支付方式 = 不合格”。现在假设我们有一个特殊的供应商（z），由于某些历史原因，他有一个合同分配权且不需要此类特定授权。可以这样描述这条规则以及它的异常：“IF 支付类型 = x，and not 授权类型 = y，and not 供应商 = z THEN 支付方式 = 不合格”。结束。

除非这种方法的使用存在一些问题。首先，此类异常可能有很多种类。如果所有这些异常都以“and not”的形式加入规则，那原本美观、简洁、通用的业务规则将会变成一团乱麻。其次，这些细节性的经营决策异常需要由不同员工在考虑不同时间尺度上潜在的战略业务策略（通常保持不变）的情况下进行管理。这种问题可以通过分离通用规则与异常条件来解决。此时策略规则变为：“IF 支付类型= x and not 授权类型 = y and not 支付授权异常 THEN 支付方式 = 不合格”。然后，异常可统一描述为一组独立的规则，包含最初的“IF 供应商 = z THEN 支付授权异常”。

但此时尚未完成所有工作。当供应商 z 未经授权便提交了一个发票，而决策服务很正确地允许它被支付时，将会发生什么？若支付授权规则在整个组织内广为人知，难道不会有人拦截此支付并将其伪装成计算机失误吗？当一个业务政策众所周知时，我们可能需要记录其异常情况，并将此作为偏离业务政策案例处理的解释。因此，我们需要把异常规则加入决策返回的理由列表中。当使用一条政策规则时，应可在理由列表中找到继续或者撤销这一政策规则的理由。

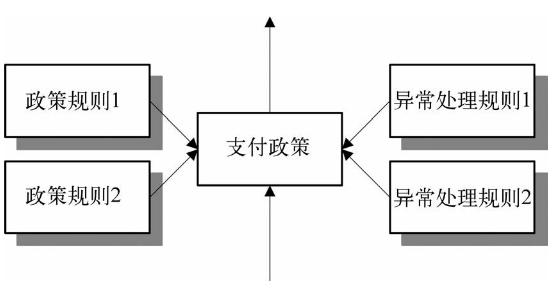


图 6-3　带异常记录的规则决策模式的 DRD 组件

2. 模式

带异常记录的规则决策模式是带理由的分类决策模式的一种拓展。如图 6-3 和图 6-4 所示。

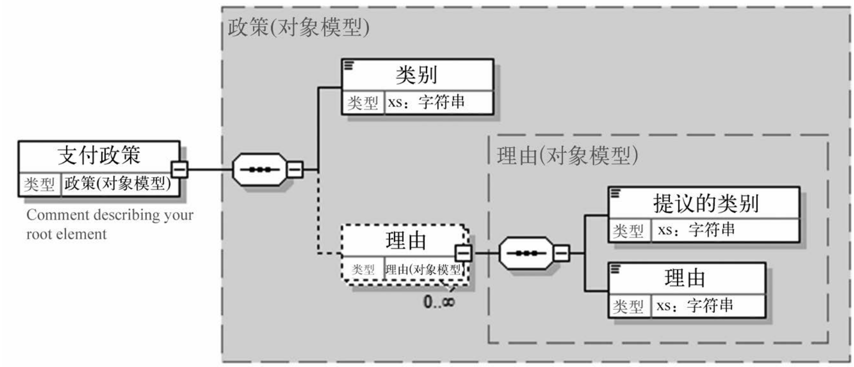


图 6-4　带异常记录的规则决策模式的对象模型

- DRD
  - 包含一个单一决策节点以及针对每个政策规则域和异常规则域的独立知识节点，如图 6-3 所示。决策的输出结果为：一个类别值及其相应的理由列表。
- 知识
  - 包括一组决策值，此时需指定其中一个决策值为默认值，同时为冲突解决方案确定一个优先级别（如第 2 章所述），通常是“拒绝” > “待定” > “接受”。
  - 包括一个或多个政策规则集，包含以“IF <条件> and not<异常> THEN 提议(<类别>，<理由>)”形式表现的规则。
  - 包括一个或多个异常规则集，包含以“IF <条件> THEN <异常>；提议(<默认类别>，<理由>)”形式表现的规则。
- 对象模型
  - 一个决策对象，带有一个类别属性及其理由列表。
  - 一个理由对象，记录了一条规则提议的类别、相关理由以及其他必要属性。

带异常记录的规则决策对象模型与带理由的分类决策模型一样（见图 6-4），此处讨论的例子不需要流通路径选择的额外信息。

- 决策基础结构
  - 包含决策流中的一个独立决策任务，其初始决策值为默认类别值，该任务随后可在相关知识域中使用其所有决策规则集。
  - 包含一个提议函数，用于实施政策规则及异常规则任务，该函数可向理由列表中增加理由，并可在提议类别比当前类别优先级更高时，更新决策类别。当被异常规则调用时，提议函数的输出结果为默认类别值，此时无需更新决策类别。
  - 一种评估异常规则的链式机制，由政策规则调用。这一机制的运行细节取决于你的 BRMS。

6.1.3 细分模型

1. 用法

如第 3 章内容所述，评分卡是一个由一组计算预测评分的规则组成的模型。然而，评分卡很少单独使用。一个评分卡模型可能被分割为多个评分卡，它使用排除规则来识别那些不能被模型覆盖的案例。通常，一个评分决策包含 3 个步骤。

(1) 通过使用一组规则，一个案例或者被排除在评分模型之外或者可被放入一个细分范围。所选细分方式决定了所需使用的评分卡类型。

(2) 某特定评分模型使用的一组特征，可通过使用计算或逻辑表示方法组合各种输入数据推导得出。例如，从婚姻状况和人口数量，可得出一个“家庭类型”特征。

(3) 应用选定的评分卡模型，将其包含的每个特征置入一个可决定局部评分的区间（特征范围）中，然后将所有的局部评分（对于总评分的贡献）加和。

逻辑上来说，这 3 个步骤都是申请风险评分决策流程的一部分，因此，可使用 DRD 表示，同时可使用决策流中一个单独的任务来执行它们。

2. 模式

图 6-5 和图 6-6 展示了细分模型的决策模式。

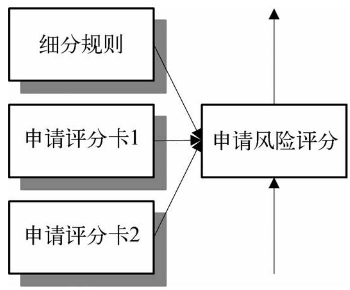


图 6-5　细分模型的决策模式的 DRD 组件

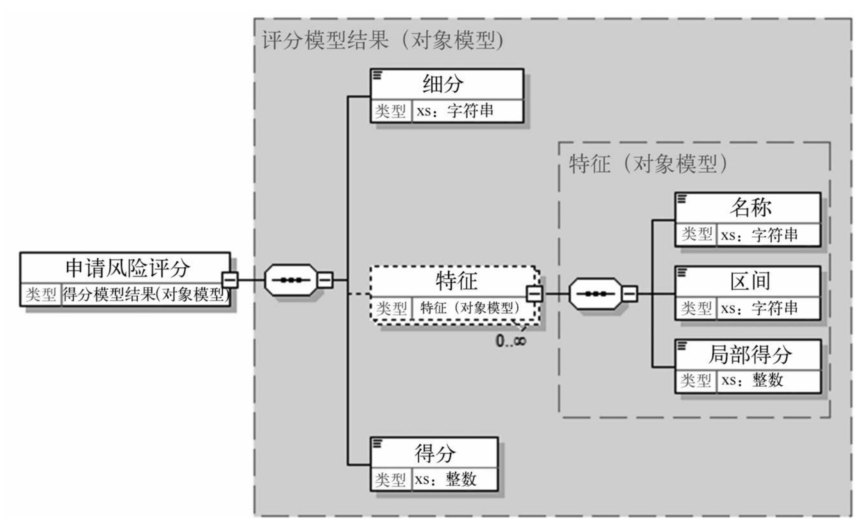


图 6-6　细分模型的决策模式的对象模型

- DRD

- 一个带有多个知识节点的单一决策节点：
  - 一个记录细分规则的知识节点
  - 每个评分卡模型各自对应一个知识节点
- 决策的输出结果包括：一种细分方式、一个评分和一个包含所有局部评分的列表。图 6-5 展示了一个带有两组细分规则的评分模型的 DRD。

- 知识
  - 一组细分规则，案例或者被置于其中一个细分中，或者被标记为排除在模型之外。
  - 每种细分都有一个评分模型，包含特征推导及评分规则等。
- 对象模型
  - 一个评分模型结果对象，带有细分属性、特征列表及最终评分。
  - 一个特征对象，记录了特征名称、置入区间以及特征的局部评分。

  图 6-6 展示了一个用于返回申请风险评分决策结果的评分模型结果对象。该对象可同时被重用于其他评分模型决策。
- 决策基础结构
  - 包含决策流中的一个独立任务，该任务使用细分规则对决策结果进行细分，然后使用合适的评分卡模型计算评分或（若不在细分范围内）使用特定符号值进行标记（如零）。

6.1.4 冠军–挑战者模型

1. 用法

任何绝对意义上来说，决策服务中表述的大多数业务知识都是错误的，它们仅仅是当前最好的实践结果。必须不断地对决策服务进行持续更新，通常可以通过在更新的分析模型、规则集等基元中表述新的业务政策来实现这一点。但是，对于一个组织来说，若不经过测试就直接在服务中更换业务政策则太过鲁莽了。冠军–挑战者模型是一种普遍用于测试业务政策有效性的方法。使用图 3-3 中所示的反馈循环架构，可使组织避免承担过度的风险。

任何时候，都可能有许多版本的业务知识可被用于制定一个决策：一个冠军和若干挑战者。每个知识版本均可被称为一个“策略”。冠军策略代表目前最好的

实践策略，挑战者策略代表其他有待测试的备选策略。为每个案例随机分配一个决策策略。大多数案例都将使用冠军策略，但是也有一小部分案例会被分配使用挑战者策略。对比那些成功执行的挑战者策略与冠军策略，统计它们一段时间内的流程反馈及后续客户行为的反馈。若统计数据表明，存在某一挑战者策略比冠军策略更好，此时，该挑战者策略成为当前的冠军策略，并再次循环。

在使用冠军–挑战者模型时，我有如下两个建议。

(1) 与每个策略相关的业务政策涵盖了不同版本的完备知识域。例如，一个完整的规则集、图表或模型。不要试图在单个规则或特征中应用冠军–挑战者模型，这将使得对结果的分析异常艰难。

(2) 使用伪随机数随机为案例分配决策策略。例如，在 0 ~ 99 中随机选取一个整数，根据这个选取的整数值（策略编码），使用一个类似于表 6-1 的图表为案例分配策略。不要只是顺序地计数并简单将每次计数的第 n 个案例分配给挑战者，这看似可取，但是由于业务流程的非随机效应（例如，工作队列中的批处理案例），会导致样本被污染。在表 6-1 中，90%的案例都被分配使用冠军策略，5%的案例使用挑战者策略 1，另外 5%使用挑战者策略 2。

注意，无论将什么策略分配给客户，都要保证一个业务流程中的每个决策点都执行该策略。因此，通常使用 BPMS 选择策略编码并将其对应策略传递到每个决策服务中。

表6-1　冠军–挑战者决策模型策略表

决　策	策略编码	策　略
申请风险级别	0～89	冠军
申请风险级别	90～94	挑战者1
申请风险级别	95～99	挑战者2

2. 模式

图 6-7 和图 6-8 展示了冠军–挑战者模型的决策模式。使用表 6-1 中展示的申请风险级别决策对该模式进行阐释，该模式适用于任何类型的决策。

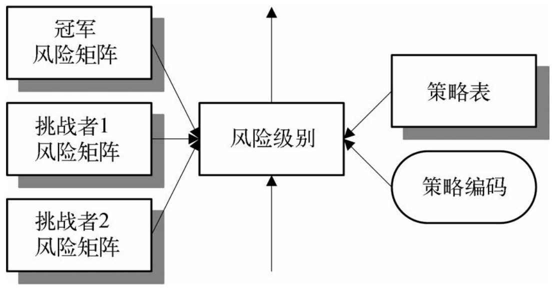


图 6-7　冠军–挑战者模型的决策模式的 DRD 元件

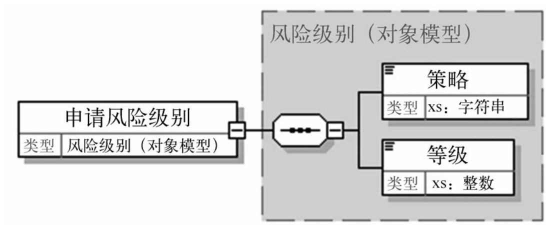


图 6-8　冠军–挑战者模型的决策模式的对象模型

- DRD
  - 一个带有多个知识节点的单一决策节点：
    - 一个策略都表知识节点
    - 每个策略都对应一个知识节点，并为其提供后续分析支持，决策结果应同时包括所用的细分模式。图 6-7 展示了一个含三个版本风险矩阵——一个冠军版本和两个挑战者版本——的风险级别决策的 DRD。
- 知识
  - 一个策略表，使用策略编码决定所用策略，该表适用于多种决策模式。
  - 每个策略所对应的一个完备的知识版本。
- 对象模型
  - 决策的标准化输出结果，包括一个策略属性，记录了所用的策略。

图 6-8 展示了一个用于返回申请风险级别结果的风险级别对象。

- 决策基础结构

- 包含决策流中的一个独立任务，该任务使用策略表决定决策所用策略，然后调用适合该策略的知识域（如图 6-7 中合适的风险矩阵）。

6.2 协同决策机制

本书刻意将关注点置于如何使用决策服务进行经营决策自动化，但是，并非任何时候都可将业务决策的责任完全交给一项决策服务，其结果也不一定能够完全令人满意，有时需要一个 BRMS 来帮助或协同用户制定决策。有两种完全不同的方法可用于协同决策制定：待定需评审以及更为微妙的决策支持。

6.2.1 待定需评审

待定需评审的原理在于，通过判定一个案例是否能够被自动化处理，或者是否必须由员工处理，对案例进行划分。若可使用清晰的规则表述这种划分逻辑，即可使用一种自动化的决策服务处理这一过程，以实现高效划分。然后，我们可使用一种完全自动化的直通式流程处理案例，而用户则只需在发生异常情况时参与处理。

第 2 章和第 5 章分别讨论了待定需评审的两种处理方式：权限分级和鉴别归类。

1. 权限分级

实现待定需评审决策的最简单的方式为：为决策服务定义一个权限等级，并将超过该等级权限的案例提交给其上层相关人员进行处理。一般情况下，这种方法包含多个审批级别，决定了某一案例具体需要哪一类型人员进行审批，如图 2-3 所示。有时需要一些业务逻辑来判断某一度量指标对应的审批等级，在业务领域，这一度量指标可能是案例潜在的销售价值、潜在风险或认购额等，然后将该值与审批表（如表 2-2 所示）中对应的权限阈值进行比较。我们可以通过使用一个加权函数输出一个总指标，或针对特定实际情况定义一些分开考虑多个度量指标的待定规则，对度量指标进行合并处理。

2. 鉴别归类

一种简单又常用的待定案例处理方法是鉴别归类。该方法现在的应用场景远不如最初的应用场景残酷，最初的场景是鉴别是否要对战场上的伤员进行救治。伤员被划分为 3 组：确定实施救治的伤员，确定不实施救治的伤员，处于两者之间需要人工审议鉴别的伤员。这正与第 3 章中描述的流通路径选择决策中的“接受”“拒绝”和“待定”相对应。

通常需使用两个规则集鉴别分类的实施：拒绝规则（明确定义了可被拒绝的案例集）和待定规则（定义了需要被人工评审的案例集）。那些不属于任何拒绝或待定集合的案例则通过一个直通车式的处理流程直接通过。注意，该系统并未明确给出一组规则，对那些应被接受的案例提供一个清晰、通用的定义。接受集合的边界在一定程度上依赖于待定案例的划分。

通过类别鉴定，决策服务和用户最终将负责不同案例。决策服务负责那些可清晰判定的案例，执行“接受”/“拒绝”决策，并将那些比较模糊的案例提交给用户。用户负责对“待定”案例进行判断，并执行“接受”/“拒绝”决策。

6.2.2 决策支持

另外一种方案是允许决策服务和用户针对单个案例展开合作，而非让决策服务或用户单独负责整个案例。该方案被称为决策支持，即假设决策服务可为用户决策提供支持（虽然，正如我们即将看到的，事情并非总是那么简单）。决策支持可用的技术内容主要依赖于用户和决策服务各自的角色，即决策责任是如何在这两个协同决策制定者之间分配的。本章将要介绍的 3 项技术内容为：

(1) 决策服务建议用户

(2) 决策服务限制用户

(3) 用户限制决策服务

它们按照逐步增加用户对于决策服务的责任的顺序进行排列。

无论使用哪一条技术内容，用户和决策服务的作用在最初即应被非常清晰地

定义，这对于项目至关重要。过多的人机交互为混乱设计的“趁虚而入”提供了一个绝佳的机会。无论决策服务在人机交互任务中发挥什么作用，它所做的贡献均可以依据决策来判定并使用 DRA 进行分析。

1. 决策服务建议用户

该技术内容使得用户在决策制定流程中有全面的灵活性及全部权限，同时使得决策服务可提出相应的意见或建议（见图 6-9）。

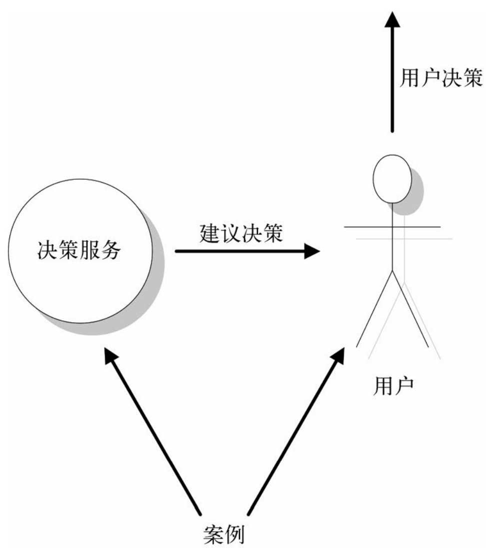


图 6-9　决策服务建议用户

注意，如果使用错误的方式来理解某一问题，例如你认为决策服务的目的是选择一份咨询报告展示给用户，就会使该过程变得异常困难。根据特定场景，选择要给出什么建议，是很困难的。因为这不仅取决于案例的特征，同时取决于用户的需求以及他们如何看待这个案例，而这类信息对于决策服务来说并不易获取。稍微不注意，就会出现上千条规则响应某一特定场景的情况，而且通常并没有简单方法来组织或维护这些规则。有效使用这种技术的关键在于，谨记决策服务提供的建议为用户需要制定什么决策，所以一种更简单的方法是让决策服务制定决策，然后将决策建议提供给用户并解释为何如此决策。

决策服务这种用法本质上等同于它在流程自动化中的用法，主要区别在于业务流程使用决策服务结果的方式不同。此处决策服务的结果仅仅展示给用户，而不是被返回到 BPMS 中并以此来影响业务流程。接下来用户将制定自己的决策，这些人为决策将被应用于流程之中。但是，此处决策服务的范围可能比决策自动化范围更广，例如，可能需要决策服务列出所有可行解，而不仅仅是选择最优解。此外，决策服务可能比决策自动化更注重决策的解释，例如，仅仅为所选决策提供一个理由列表有时可能并不足够，有时需提供一个完整的决策逻辑过程。

2. 决策服务限制用户

在该技术内容架构下，用户需对案例的决策制定流程负责，但是用户的行为受到决策服务的限制，后者有责任保证用户的决策是前后一致的且符合法律和政策规定。

根据决策服务施加于用户的限制的复杂程度，我们有两种可行的但是大相庭径的方案：前摄式的和反应式的。

(1) *前摄式限制*。若限制十分简单并可独立地应用于个体用户的决策值，那么决策服务即可对用户将要做出的每个决策施加合适的限制，并可通过用户界面或 BPMS 的使用确保用户在这些限制下制定前后一致的决策。如图 6-10 所示。这种方法的常见应用包括：使用决策服务设置数据输入字段允许接受的取值范围（例如，收费的上下限），或者为下拉列表（例如为用户提供客户可选的产品）提供一组可能取值。图 6-2 中“拒绝是可改写的”标记即为一个对用户后续决策施加限制的例子。

为实现 DRA 的目标，决策服务将要执行的主要决策是判定这些限制的取值，而非执行用户做出的后续决策。

(2) *反应式限制*。如果希望对用户决策之间的联系实施限制，例如确保一组决策是一致的，就必须使用与前摄氏限制相反的方法。反应式限制的步骤为：用户先制定一些决策，然后决策服务会检查这些决策是否符合限制要求。图 6-11 展示了这一过程。图中，决策服务的输入为用户提议的决策，服务对这些决策是否符

合限制要求进行判断，并将不符合限制要求的决策提议返回给用户，同时附上解释说明。用户调整其提出的决策直至决策被服务批准。

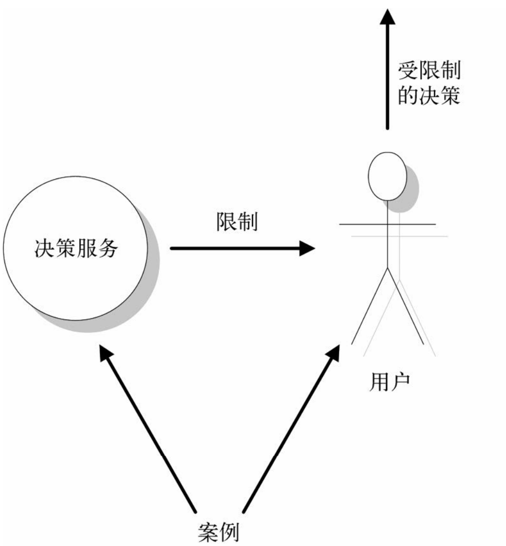


图 6-10　决策服务限制用户——前摄式限制

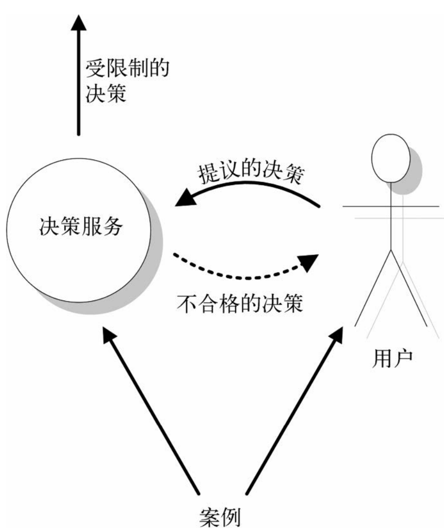


图 6-11　决策服务限制用户——反应式限制

3. 用户限制决策服务

使用该技术内容进行协同决策制定时，所有决策制定流程均由决策服务执行，但是受到用户的限制。这种用户限制常被称为“改写”，图 6-12 展示了这个过程。当出现以下这种情况时，就可使用这种用户限制决策服务的技术：当决策之间相互关联且结构复杂，而你希望用户能够有权改写结构中的单个组件的决策，但是又希望决策服务可以确保决策的整体结构是正确且一致的。此处谨记一条十分重要的原则：永远不能改写规则，只能改写决策。

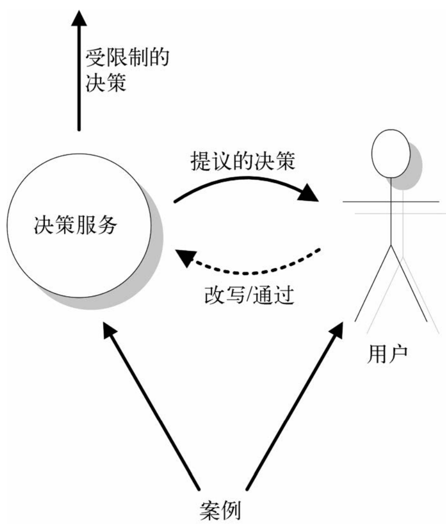


图 6-12　用户限制决策服务

保险索赔裁决系统是使用这种技术的一个很好的例子，其中索赔内容可能由许多费用条目组成，并且系统中有许多复杂的规则来管理这些费用条目的组合，以及如何对费用条目进行加和从而获得最终赔偿总额。随后裁决者应该通过改写一个或多个局部费用条目，对赔偿总额进行酌情量裁。接下来，决策服务将优先考虑裁决者的决策，并尽可能使用裁决者给出的局部费用重新计算赔偿总额。

最简单的方法就是将决策服务所实施的决策的整体结构展现给用户，并准许用户改写任一局部决策。决策服务可随后使用原始案例数据加上改写后的数据再次被调用，这些数据都应当使用完全相同的对象模型来表示。决策服务以这些数

据为输入，输出决策结果。既然决策服务会评估其输出的每个结果，首先它应该检查某一决策值是否已由用户提供（可能以改写的方式写入）。如果已提供，那么决策服务只需单纯地将它复制到该局部决策结果的对应位置。如果没有提供，那么决策服务将继续使用原来的方式评估决策结果。

这种用户与决策服务之间的互动可不断被重复，在此过程中，用户会改写不同局部决策并观察整体效果，直到用户对结果满意并且批准决策为止。这是决策支持中最复杂的技术，但它使系统得以使用用户和决策服务协同工作提供的最优特征，这实现了真正的协作。

总　　结

通过这 6 章的学习，我们完整地经历了一个较高层次的知识自动化项目的全过程：从最顶层的激励业务的策略，到底层用于执行决策服务中业务知识的各种技术等。总结来说，本书的主要内容如下。

第 1 章使用一个基本的事实作为开端——业务知识有一种可度量的价值，其价值可通过使用决策管理的相关方法自动化业务流程中的经营决策机制来实现，并可通过决策收益定价法进行衡量。

第 2 章解释了如何使用业务流程中的一组决策点对经营决策的制定流程进行建模，以及如何使用封装了决策所需的业务知识的决策服务来自动化决策制定流程。这些服务由业务规则管理系统（BRMS）公布，并由业务流程管理系统（BPMS）调用。若要高效的使用决策服务，可能还会涉及为合理化决策点而进行的某些流程的重设计，此处描述了一个自动发放贷款的业务流程模板作为示例。

第 3 章展示了在决策服务中封装知识时使用的最重要的技术——业务规则（包括决策表及决策树），算法和预测分析方法（包括归纳法、评分卡模型以及神经元网络），并讨论了它们的相对优缺点。此处展示了自动化决策流程的整体架构，以及支持管理决策周期的 4 个重要架构组件（BPMS，BRMS，预测分析模型和数据仓库）。

第 4 章介绍了决策需求分析（DRA）方法，解释了使用 DRA 的目的，并展示了 DRA 的基本原则。同时本章提出了一种简单的研讨会技术 DRAW，DRAW 可将决策制定流程分解为能被决策需求图表（DRD）刻画的网络结构。DRD 展现了

决策、知识域和数据域之间的关系，并允许我们以此为基础界定项目范围。

第5章展示了使用DRD结构组织一个知识自动化项目的所有要点：范围确定、服务评估、项目规划、知识发现、设计、开发、配置和测试。本章内容表明，使用 DRA 对决策制定流程进行模块化，即可创建高效运作的知识生产线。

第 6 章讨论了一些具体的常用决策模式示例，同时讨论了如何在需要决策服务及业务工作人员共同参与制定决策时，使两者协同合作实现决策制定。

这就限制了我们可在本书中进行讨论的细节，我已竭尽全力在此限制下保持背景知识的全面性。再进一步就该讨论针对单个 BRMS 产品、设计工具或架构环境的技术问题。若想了解这些细节，可查阅相关产品评论及咨询合同供应商。

在本书中，我试图传达一种看待事物的方式，一种构建项目的方式，甚至是一种语言，而非展示一个被称为理论的东西。DRA 的原理非常简单，甚至是超乎异常地简单，但是，我相信这种简单并不会使其贬值。其关键理念在于：决策制定流程有一个可被图形化表示的结构，若能获取该结构，就能够以此为支撑执行一个知识自动化项目中的所有工作。其效益在于更加良好的通信、更清晰的范围、更高效的信息传递以及更低的风险。

即使不完全接受 DRA，仅采用其语言也是大有裨益的：注意仔细区分决策点、决策服务、决策、知识域、规则、算法和模型的差异。本书太过频繁地使用“规则”这一表述，可能会引起一些困惑。

因此，总结来说，对于成功的知识自动化项目，我有 10 个小贴士（我犹豫要不要使用戒律来称呼它们）。

(1) 使用决策收益定价法为可能的知识自动化项目进行审查评估。

(2) 不要自动化已存在的流程，重新设计业务流程并将其自动化。

(3) 清晰地识别业务流程中的决策点，这些决策点仅执行自动决策制定。

(4) 即使使用决策服务作为决策支持，也要依据决策制定流程对其进行逐一分析。

(5) 定义那些将使用 DRAW 进行自动化的决策制定流程，并利用 DRD 中展示的决策结构告知其所有后续活动。

(6) 通过在 DRD 上刻画边界将其划分为不同子图，确定项目自动化的范围以及单个决策服务的功能。

(7) 通过将 DRD 分解为不同功能增量并确定增量间的依赖性，构建一个迭代项目的开发计划。

(8) 通过将知识发现限制在将被自动化的每个增量的知识域中，引导知识发现。

(9) 基于 DRD 对决策流、对象模型以及每个决策的可重用实施类型进行设计。

(10) 创建一条可以从项目交付到初始需求进行全程追踪的知识生产线。

我真心希望本书介绍的自动化管理方法能够对你有所帮助，并祝你能够在每个知识自动化项目中都获得成功。

术语表

本体论 见“对象模型”。

单元测试 单元测试发生在组件集成和用户测试之前，用于证明某一功能组件可依设计正常运行。

迭代 迭代是迭代式项目管理（如 RUP）中运行的项目的一个部分，一次迭代交付项目的一个功能增量。

迭代式项目管理（IPM） 迭代式项目管理通过将项目分解为一系列小项目或迭代单元，每次迭代都强调软件的整个生命周期过程——需求、设计、构建及测试——以期实现一次功能的增加。

对象管理组织（OMG） 对象管理组织是负责 BPMN 和 SBVR 标准制定的国际协会。

对象模型 对象模型是一种使用实体知识领域及其相互关系来表示实体的结构化语言，亦被称为事实模型或本体论。

多层感知机（MLP） 多层感知机是神经元网络的一种常用形式，由一个输入层、多个隐含层和一个输出层组成。

Fair Isaac 统一软件开发过程（FIRUP） Fair Isaac 统一软件开发过程基于 RUP 的项目管理方法，在 FICO 中被广为使用。

反向传播（BP）算法 反向传播算法是一种用于调节神经元网络中连接权重从而最小化网络预测误差的方法。

封装 封装是 SOA 中采用的一种方法，它对服务消费者隐藏服务内在逻辑，

使其只能看到封装的结果。

冠军–挑战者方法　冠军–挑战者方法是一种测试新业务政策有效性的方法，该方法可避免将组织置于过度的风险中。具体做法为对大部分测试集仍使用当前冠军策略，对随机挑选的少部分测试集使用挑战者策略，并使用统计学方法比较两种策略的有效性。

归纳法　归纳法是一种从特定观察结果中推导出一组通用规则的推理形式。归纳法可用于从案例数据中自动生成通用规则与决策树。

归纳逻辑编程（ILP）　归纳逻辑编程是一种机器学习技术，使用逻辑归纳与概括的方法生成一组与给定事实相一致的规则。

规则发现　见“知识发现”。

规则集　规则集是制定某一决策所需的一组业务规则。

规则隐喻　规则隐喻是规则集的一种表示方法，它使规则更易于被业务用户理解，如表示为决策表或决策树的形式。

互斥　互斥是规则集的一种属性，指对任一输入数据集仅有一个适用规则。

机器学习　机器学习是指使用预测模型从数据中创造知识的一系列技术。

基于知识的系统　基于知识的系统是一个可对业务知识（如业务规则）进行建模并将其部署于决策服务中的系统。

集成测试　集成测试用于检测一个软件解决方案的各个组件是否能协同工作并通过接口正常交互。

继承　继承是对象模型的基本法则之一，即任一对象的属性同时是其同一类型、子类或实例对象的属性。比如，咪咪（我的猫）具有一只猫的所有属性，类似的，一只猫具有哺乳动物的所有属性。

决策　决策是指使用业务知识从一组数据中得出一组结果的判定过程。

决策表　决策表是一种规则隐喻，它将一个规则集表示为一个表格的形式。表格中的每列表示某一属性在规则条件中的取值，另一列指代对应条件下得到的结论。表中的每一行代表一条业务规则。

决策点　决策点是业务流程调用决策服务实施一个或多个决策的点。

决策服务 决策服务以一组描述案例的数据作为输入，通过应用一些已封装的业务知识，返回一个或多个针对该案例的决策。

决策管理（DM） 决策管理是指使用机器学习以及自动化决策机制来提高业务决策的盈利能力，亦称为企业决策管理。

决策流程规范 业务流程规范是由 Paul Konnersman 发明的一种业务流程建模技术，这一技术将决策建模为涉及 5 种角色（决策管理者、咨询顾问、审批者、检查员和通讯员）的活动。

决策收益定价法 决策收益定价法是一种用于预测、衡量决策管理项目投资回报率的方法，它由 FICO 开发，通过五方面指标（精度、成本、速度、敏捷性和一致性）评价决策管理带来的收益。

决策树 决策树是一种规则隐喻，它将一个规则集表示为可对案例进行分类的树的形式。树中每条可能的路径表示一条业务规则，树中路径上的每个节点表示一个条件，而终端节点表示案例所属的可能类别。

决策需求分析（DRA） DRA 是一种决策流程的分解分析方法，通过定义各个决策所需的信息，如业务知识领域、数据域以及其他决策的结果等，对决策流程进行分解。其结果以 DRD 的形式呈现。

决策需求分析研讨会（DRAW） DRAW 是一种结构化的研讨会技巧，可为一系列决策服务定义决策需求，并以 DRD 的形式将之存档。

决策需求图表（DRD） DRD 是一个网络图，它将一个决策制定的领域描述为由决策、子决策及其相关支持业务知识和数据组成的网络结构图。

面向服务的体系结构（SOA） SOA 是一种软件组件模型，它使用可重用、松耦合的功能组件来封装内部逻辑，使内部逻辑对用户透明。

目标导向推理 见逆向链式推理。

逆向链式推理 逆向链式推理是一种基于规则的系统推理策略，该方法首先假设目标结论成立，然后对使结论成立的必要规则进行评估。这一策略也被称为目标导向推理。

评分卡模型 评分卡模型是一个预测模型，它通过对一系列加权特征进行加

总并产生一个数值评分，以此来预测一个具体事件发生可能性，可用于申请评分、行为评分和信用评分，亦被称为评分模型。

瀑布式方法 瀑布式方法是项目管理的一种方法。该方法规定，每次开始软件生命周期新一阶段——需求分析、设计、构建、测试——前必须先完成上一阶段的开发。

企业决策管理（EDM） 见决策管理

RETE 算法 RETE 是一种用于优化业务规则引擎推理策略的算法。

商务智能（BI） 商务智能是指一组整合、分析、过滤及表示数据的技术，它能够协助决策执行者制定决策，而非替代执行者直接执行决策。

神经元网络 神经元网络是一种通过模拟简化的神经元细胞网络的机理来实现机器学习的方法。它通过调整神经元间的链接权重来实现学习的过程。

事实模型 见“对象模型”。

数据导向的推理 见“正向链式推理”

算法 算法是一种程序，它可根据一组输入数据计算得出结果，通常表现为一个计算步骤的序列，通常在软件中作为函数来实现。

统一软件开发过程（RUP） RUP 是一种迭代式项目管理方法，该方法将项目划分为 4 个阶段：初始阶段、细化阶段、构造阶段和交付阶段。

推理 推理是运用业务规则从一组数据中得出一个结论的过程。

推理引擎 见“业务规则引擎”。

完备性 完备性是规则集的性质之一，能够为输入数据项的所有可能组合定义其决策结果。

伪代码 伪代码是一种非正式但结构化的语言，近似于一种非特定的编程语言，可用于解释算法流程。

细分 细分是一种通过将案例总体切分为若干部分并针对每部分单独建模以提高模型预测精度的技术，细分常用决策树的形式表示。

业务规则 业务规则是组成业务知识的“原子”，表现为：IF <条件> THEN <结论>的形式。

业务规则管理系统（BRMS） BRMS 是构建决策系统的软件环境，由业务知识库、业务规则引擎、开发环境、用于创建知识维护接口的组件以及部署决策服务的工具等组件构成。

业务规则引擎（BRE） BRE 是 BRMS 中执行业务知识（包括业务规则）的组件，又称推理引擎。

业务流程管理（BPM） 业务流程管理是一门关于业务流程设计与实现的专业学科，目的是使其对组织的价值最大化。

业务流程管理系统（BPMS） 通过执行业务流程模型中定义的活动序列来管理业务流程的计算机系统。

业务流程建模符号（BPMN） 业务流程建模符号是业务流程模型的一种标准表示形式，遵循这一标准的模型可直接被业务流程管理系统执行。

业务流程模型 描述某业务流程的活动组成以及活动之间的依赖关系的流程图表。

一致性 一致性是规则集的一个性质，它是指对任何一种情况，或者仅有一个规则被触发，或者多个规则被触发但存在判定正确输出结果的方法。

用户测试 用户测试用于阐明客户满意度：交付系统满足客户所有需求。

由外至内 由外至内的流程设计方法从期望的客户服务出发，提供客户服务所需的业务流程。

预测分析建模 预测分析建模根据一系列已发生的典型案例提供的历史数据进行分析，继而创建用于客观预测未来可能发生的类似案例的模型。

预测模型 预测模型运用预测分析技术建立数学模型，用于预测给定数据集的对应结果。

增量 增量为迭代式项目管理（如 RUP 等）中一次迭代所增加的功能模块。

正向链式推理 正向链式推理是基于规则的系统中的一种推理策略，在该策略中，BRE 从已知数据出发，确定可由该数据集得出的所有结果。也称为数据导向的推理。

知识发现 知识发现是指从领域专家处收集业务知识，并将其编写为可执行

的方式（例如业务规则或算法）。亦被称为**规则发现**。

知识自动化 使用决策管理、业务流程管理系统和面向服务的架构，在业务流程中实现决策自动化。

自动化范围文档 自动化范围文档包含了 DRAW 的结果，它用于流程自动化项目中，可界定将被自动化的决策制定流程的范围。

延展阅读

书籍及文章

Davenport, T. A., J. G. Harris, and R. Morison. *Analytics at Work: Smarter Decisions, Better Results*. Boston: Harvard Business School, 2010.

Debevoise, T. *Business Process Management with a Business Rules Approach: Implementing the Service Oriented Architecture*. Lexington, VA: Tipping Point Solutions, 2007.

Debevoise, T. and R. Geneva. *The Microguide to Process Modeling in BPMN: How to Build Great Process, Rule, and Event Models*. Lexington, VA: Tipping Point Solutions.

Fausett, L. *Fundamentals of Neural Networks: Architectures, Algorithms and Applications.* Upper Saddle River, NJ: Prentice-Hall, 1994.

Gilovich, T., D. W. Griffin, and D. Kahneman. *Heuristics and Biases: The Psychology of Intuitive Judgment*. New York: Cambridge University Press, 2002.

Graham, I. *Business Rules Management and Service Oriented Architecture: A Pattern Language.* Chichester, UK: John Wiley & Sons, 2006.

Harmon, P. *Business Process Change: A Guide for Business Managers and BPM and Six Sigma Professionals,* 2nd edition. Burlington, MA: Morgan Kaufmann Publishers, 2007.

Konnersman, P. M. *Decision Process Specification: A Process for Defining Professional and Managerial Work Processes*. Portland International Conference on Management of Engineering and Technology, July 27–31, 1997, Portland, Oregon.

Kruchten, P. *The Rational Unified Process: An Introduction,* 3rd ed. Boston: Addison-Wesley, 2003.

Morgan, T. *Business Rules and Information Systems: Aligning IT with Business Goals*. Boston: Addison-Wesley, 2002.

Rational, *Rational Unified Process: Best Practices for Software Development Teams*. Rational Software White Paper TP026B, rev. 11/01, 2001.

Rosenberger, L. E., and J. Nash. *The Deciding Factor: The Power of Analytics to Make Every Decision a Winner*. San Francisco: Jossey-Bass, 2009.

Ross, R. G. *Principles of Business Rule Approach*. Boston: Addison-Wesley, 2003.

Ross, R. G. *Business Rule Concepts: Getting to the Point of Knowledge*. Houston: Business Rule Solutions, LLC, 2009.

Ross, R. G. and G. S. W. Lam. *Building Business Solutions: Business Analysis with Business Rules*. International Institute of Business Analysis handbook. Houston: Business Rule Solutions, LLC, 2011.

Taylor, J. *Smart (Enough) Systems: How to Deliver Competitive Advantage by Automating Hidden Decisions*. Boston: Pearson Education Inc., 2007.

Taylor, J. *Decision Management Systems: A Practical Guide to Using Business Rules and Predictive Analytics*. IBM Press, 2011.

Von Halle, B., L. Goldberg, and J. Zachman, *Business Rule Revolution: Running Business the Right Way*. Silicon Valley: Happy About, 2006.

Von Halle, B. *Business Rules Applied: Building Better Systems Using the Business Rule Approach*. New York: John Wiley & Sons,2001.

Von Halle, B. and L. Goldberg,. *The Decision Model: A Business Logic Framework Linking Business and Technology*. Boca Raton, FL: Auerbach Publications, 2010.

网站及博客

The Business Rules Manifesto: The Principles of Rule Independence. (Ed. Ronald Ross). www.businessrulesgroup.org/brmanifesto.htm

BPM Focus. http://www.waria.com

Business Process Management Initiative. http://www.bpmi.org

Business Process Trends. http://www.bptrends.com

Business Rules Community. http://www.brcommunity.com

Business Rules Group. http://www.businessrulesgroup.org

Complex Event Processing (blog). (Ed. Paul Vincent). http://tibcoblogs.com/cep

Decision Management (blog). (ed. James Taylor). http://www.ebizq.net/blogs/decision_management

Everything Decision Management, Technically Speaking (blog). (Ed. Carole-Ann Matignon and Carlos Serrano-Morales). http://techondec.wordpress.com

FICO Decision Management (blog). http://dmblog.fico.com

FICO Decision Management Community. http://exchange.fico.com/dmc

International Institute for Analytics. http://iianalytics.com

Wanderings of the Mind: JavaRules (blog). (Ed. James Owen). http://javarules.blogspot.com.

Jim Sinur (blog). http://blogs.gartner.com/jim_sinur

JT on EDM (blog). (Ed. James Taylor). http://jtonedm.com

Object Management Group. http://www.omg.org

Primatek Consulting (blog). (Ed. Eric Charpentier). http://www.primatek.ca/blog

Process Modeling 2.0 (blog). (Ed. Tom Debevoise). www.tomdebevoise.com

Requirements Networking Group. http://www.requirementsnetwork.com

Sparkling Logic Community. http://community.sparklinglogic.com

关于作者

Alan Fish 是使用业务规则和预测分析进行决策管理领域的权威专家。他在该领域的创新包括决策服务分析、设计和开发，以及决策需求分析等技巧方法论。Alan 在心理学、统计学和人工智能等学科方面的背景使得他能够将人类知识、预测分析和业务规则融入决策制定体系中。

Alan 从事决策系统的构建工作已有 30 余年，并主持了许多当时的技术前沿重大项目。在这期间，他参与了许多不同领域的工作，包括国防、机器人技术、工业过程控制、政府、公共事业资产管理、保险及零售信贷发放等。

Alan 现为 FICO 决策方案部门的首席顾问，工作地点主要在欧洲、中东和非洲。在此之前，他在一家小型咨询公司任常务董事，该公司主要从事面向基于规则的系统应用及数据挖掘等工作。他在这个公司工作了 12 年之久。再之前，他作为一名人工智能和人机工程领域的咨询师，受雇于多个国际 IT 系统公司。他先后获得了曼彻斯特大学心理学专业硕士学位和神经网络专业的博士学位。

目前，Alan 和他的妻子以及两个年幼的孩子住在他亲自搭建的房子里，这座房子位于刘易斯岛（外赫布里底群岛，苏格兰），毗邻海边，他在那里养了一些羊和鸡。他还有四个成年的孩子。他对各种形式风格的音乐都有浓厚的兴趣，他和古典唱诗班、无伴奏组合、摇摆乐队一起唱歌；他同时是一个吉他手和一个创作歌手；他编排现代古典艺术作品以及爵士乐队作品；他是萨克斯的狂热爱好者。除此之外，他也喜欢散步、阅读、做饭，特别是在威士忌以及烟熏味的陪伴下与朋友们聊天逗乐。

关于译者



王飞跃教授，1990 年获美国伦塞利尔理工学院（RPI）计算机与系统工程博士学位。1990 年起在美国亚利桑那大学先后任助理教授、副教授和教授。1998 年作为国家计划委员会“引入海外杰出人才计划”和中国科学院“百人计划”人才回国工作，2011 年追溯为国防领域“千人计划”国家特聘专家。曾任中国科学院自动化研究所副所长，现为中国科学院自动化研究所复杂系统管理与控制国家重点实验室主任，国防科技大学军事计算实验与平行系统技术研究中心主任，中国科学院大学中国经济与社会安全研究中心主任，青岛智能产业技术研究院院长。

主要研究领域为智能控制、社会计算、平行系统、知识自动化等。自 20 世纪 80 年代起从事自动化和复杂系统的研究与应用，提出并建立了智能系统的协调结构和理论、语言动力学理论、代理控制方法、复杂系统的 ACP 方法等。自 21 世纪初，发起并开拓了社会计算、社会制造、平行控制、平行管理等新的研究领域。

曾任 *IEEE Intelligent System* 主编及多份 IEEE 和其他国际期刊的主编、副主编或编委，现任 *IEEE Transactions on Intelligent Transportation System* 和《自动化学报》及《指挥与控制学报》的主编。曾任 IEEE 智能交通系统学会主席（2007–2008）、旅美科协主席（2005）、北美竺可桢教育基金会会长（2006–2008）。现任中国自动化学会副理事长兼秘书长。2007 年获国家自然科学二等奖和 ACM 杰出科学家称号，2003 年起先后当选 IEEE、INCOSE、IFAC、ASME 和 AAAS 等国际学术组织的会士。2014 年获诺伯特・维纳奖。

版权声明

“Fish博士认为，决策需求的分析是在业务流程工作中实现知识自动化的关键，也是企业决策管理的核心所在。通过使用决策需求分析图表，将决策、知识和数据之间的依赖关系图形图像化，清晰地阐明‘局部知识用于局部决策，全局知识用于全局决策’的‘所要即所需，所得即所用’理念，更好地实现知识的价值。本译著的出版对于从事知识工作的自动化的研究具有重要参考价值。”

——柴天佑，东北大学教授，中国工程院院士

“2013年麦肯锡发布了题为《颠覆技术：即将变革生活、商业和全球经济的进展》的全球报告，预测了12项可能在2025年之前决定未来经济的颠覆性技术，其中第二名便是以智能软件系统为代表的‘知识工作的自动化’，并预估该技术可在2025年带来5.2万亿至6.7万亿美元的经济效益。Fish博士从企业经营管理的角度出发，探讨了如何获取知识以及如何将知识应用于企业决策制定等问题。但他的方法并不局限于企业的应用，同样适合于工业流程中的决策制定与管理。本书条理清晰，令人受益匪浅。”

——桂卫华，中南大学教授，中国工程院院士

“‘知识自动化’是传统人工智能的基石，Fish博士强调了知识在业务流程中的应用及价值体现，借助规则、算法及预测分析模型等方法，将知识封装在决策服务中，并在面临特定需求的时候调用并使用知识。‘企业知识自动化’只是人工智能‘知识工程’大概念中的沧海一粟，他的方法却为人工智能的知识驱动法提供了一条可行的途径。”

——张钹，清华大学教授，中国科学院院士

“培根的名言‘知识就是力量’照耀了人类社会进步五百多年。物质运动的自动化已近百年，显著地改变了人类生产与探索世界的进程。当今，信息科技是建立在数据之上的，信息科技的巨大发展必然导致大数据的时代，这个时代呼唤知识的自动化。现在讨论这个问题正逢其时！”

——黄琳，北京大学教授，中国科学院院士

“知识自动化是信息自动化在开放的网络空间中的延伸与提升，它并非只是知识的自动产生过程，更能够诱发知识的传播、获取、分析、影响、产生等方面的重要变革。”

——吴宏鑫，中国空间技术研究院研究员，中国科学院院士

“Fish博士提出的收集、建模与管理决策需求的方法行之十分有效。对于任何想要建立决策管理系统，或是高效地以决策为中心进行形势分析和开展商业自动化的组织、团体或个人，本书都是必读之选。”

——James Taylor，Decision Management Solutions公司CEO、首席顾问

“Fish博士具有IT、人工智能、基于规则系统开发的复合背景。本书中，他着重解决了两个问题：其一是基于规则系统在辅助决策中的应用，其二是将这些应用自动化。对于关注这些主题的企业管理者和分析师，特别是对如何在商业流程中使用规则系统的读者，本书提供了一个完整的概述，且文笔流畅、易于理解。”

——Paul Harmon，Enterprise Alignment公司创始人、首席顾问

WILEY

Copies of this book sold without a Wiley sticker on the cover are unauthorized and illegal.

图灵社区：iTuring.cn

热线：(010)51095186转600

分类建议	计算机/信息管理
	经济管理/决策制定

人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-40261-5

9 787115 402615

ISBN 978-7-115-40261-5

定价：49.00元

看完了

如果您对本书内容有疑问，可发邮件至 contact@turingbook.com，会有编辑或作译者协助答疑。也可访问图灵社区，参与本书讨论。

如果是有关电子书的建议或问题，请联系专用客服邮箱:
ebook@turingbook.com。

在这可以找到我们:

微博 @图灵教育：好书、活动每日播报
微博 @图灵社区：电子书和好文章的消息
微博 @图灵新知：图灵教育的科普小组
微信 图灵访谈：ituring_interview，讲述码农精彩人生
微信 图灵教育：turingbooks